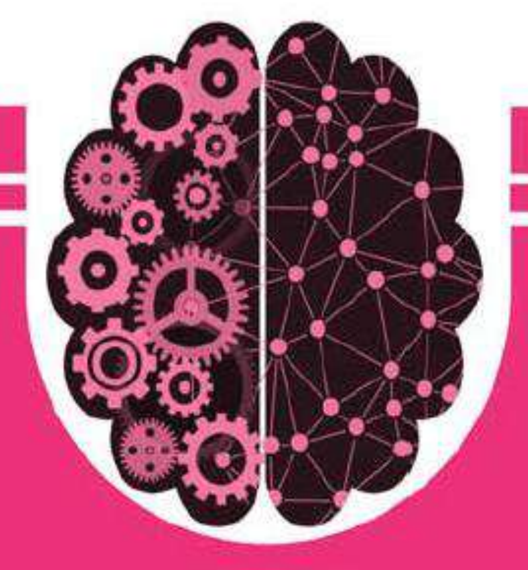


التعلم العميق

من الأساسيات حتى بناء شبكة عصبية عميقة بلغة البايثون

تأليف: ميلاد وزان

ترجمه: د. علاء طعيمه

بســمه تعالى

التعلم العميق

من الأســاسيات حتى بناء شبكة عصبية عميقة بلغة البايثون

تأليف:

ميلاد وزان

ترجمة:

د. علاء طعيمة


mohamed khatab

مقدمة المترجم

يقدم هذا الكتاب دروساً تعليمية عديدةفي مجال التعلم العميق ويتميز الكتاب ببساطة لغته وسهولة فهمها من قبل القارئ مع شرح مميز مدعم بالأمثلة والتمارينفي نهاية كل فصل.

عند انتهائي من قراءه هذا الكتاب، احببت ان اترجم هذا الكتاب وبعد التواصل مع المؤلف الاستاذ ميلاد وزان لم يبد مانعا من ترجمته الى العربية. ولله الحمد ترجمت الكتاب الى العربية وحسب معلوماتي لا يوجد كتاب عربي يتناول التعلم العميق.

لقد اخترت كتاب "يادگيری عميق" للأستاذ ميلاد وزان لما رأيته من جودة هذا الكتاب، وللمنهجية التي اتبعها المؤلففي ترتيبه وبساطة شرحه. لقد حاولت قدر المستطاع ان اخرج بترجمة ذات جودة عالية، ومع هذا يبقى عملاً بشرياً يحتمل النقص، فاذاكان لديك أي ملاحظات حول هذا الكتاب، فلا تتردد بمراسلتنا عبر بريدنا الالكتروني alaa.taima@qu.edu.iq .

نأمل ان يساعد هذا الكتاب كل من يريد ان يدخلفي مجال التعلم العميق ومساعدة القارئ العربي على تعلم هذا المجال. اسأل الله التوفيقفي هذا العمل لأثراء المحتوى العربي الذي يفتقر أشد الافتقار إلى محتوى جيد ورصينفي مجال الذكاء الاصطناعي وتعلم الالة والتعلم العميق. ونرجو لك الاستمتاع مع التعلم العميق ولا تنسونا من صالح الدعاء.

د. علاء طعيمة

كلية علوم الحاسوب وتكنولوجيا المعلومات

جامعة القادسية

العراق

مقدمة المؤلف

التعلم العميق هو تقنية جديدة قوية تزداد شعبيتها يومًا بعد يوم في مختلف المجالات. لذلك، من المهم جدًا أن نتعلمها. هذا الكتاب مخصص للمبتدئين الذين ليس لديهم معرفة بالتعلم العميق لإعداد القراء لدورة فائقة السرعة في التعلم العميق. توقعنا الوحيد من القراء هو أن لديهم بالفعل مهارات البرمجة الأساسية في بايثون.

يهدف هذا الدليل المختصر إلى تزويدك كمبتدئ بفهم للموضوع، بما في ذلك الخبرة العملية الملموسة في تطوير النموذج. إذا كنت بالفعل فوق مستوى المبتدئين، فهذا الكتاب ليس لك!

ميلاد وزان

كاشمر-2022

المحتويات

1

مقدمة في التعلم العميق

اهداف التعليم:

- ما هو التعلم العميق؟
- سبب الإقبال على التعلم العميق
- اختلاف التعلم العميق عن التعلم الآلي
- مستقبل التعلم العميق
- تطبيقات التعلم العميق

مقدمة

التعلم العميق هو مجموعة فرعية من التعلم الآلي الذي يركز على استخدام الشبكات العصبية لحل المشكلات المعقدة. اليوم، أصبح أكثر شيوعًا بفضل التطوراتفي البرامج والأجهزة التي تسمح لنا بجمع ومعالجة كميات كبيرة من البيانات. لأن الشبكات العصبية العميقة توفر كميات كبيرة من البيانات للأداء الجيد الذي نتوقعه، وبالتالي تتطلب أجهزة قوية لمعالجة هذه الكمية الكبيرة من البيانات.

ما هو التعلم العميق؟

الذكاء الاصطناعي هوفي الأساس محاكاة للبشر وسلوكياتهم العقلية بواسطة برنامج كمبيوتر يمكنه القيام بأشياء تتطلب عادة ذكاء بشري. بعبارات أبسط، نظام يمكنه محاكاة السلوك البشري. تشمل هذه السلوكيات حل المشكلات والتعلم والتخطيط التي يتم تحقيقها من خلال تحليل البيانات وتحديد الأنماط بداخلها من أجل تكرار تلك السلوكيات.

الكود أو التقنية أو الخوارزمية أو أي نظام يمكنه محاكاة فئة الفهم المعرفي التي تظهرفي حد ذاتها أوفي إنجازاته هو الذكاء الاصطناعي. يتضمن ذلك التعلم الآلي، حيث يمكن للآلات التعلم من خلال الخبرة واكتساب المهارات دون تدخل بشري. ومن ثم، فإن الذكاء الاصطناعي هو لبنة بناء التعلم الآلي.في الواقع، يعد التعلم الآلي مجموعة فرعية رئيسية من الذكاء الاصطناعي ويمكنه تمكين الآلات من استخدام الأساليب الإحصائية لجعل تجاربهم أكثر جودة ودقة. يسمح هذا لأجهزة الكمبيوتر والآلات بتنفيذ الأوامر بناءً على بياناتهم وتعلمهم. تم تصميم هذه البرامج أو الخوارزميات لمعرفة المزيد بمرور الوقت والتكيف مع البيانات الجديدة.

الفكرة الرئيسية لاختراع التعلم الآلي القائم على العينة هو أن عملية التفكيرفي المشكلة أصبحت ممكنة من خلال الإشارة إلى أمثلة مماثلة سابقة. تسمى الأمثلة السابقة المستخدمة لبناء القدرات أمثلة تعليمية، وتسمى عملية القيام بذلك التعلم.في أنظمة الكمبيوتر، هناك خبرةفي شكل البيانات، والمهمة الرئيسية للتعلم الآلي هي تطوير خوارزميات التعلم التي تقوم بنمذجة البيانات. من خلال تغذية البيانات التجريبية إلى خوارزمية التعلم الآلي، نحصل على نموذج يمكنه عمل تنبؤاتفي الملاحظات الجديدة.

التعلم العميق هو أيضًا مجموعة فرعية من الذكاء الاصطناعي والتعلم الآلي حيث تكتسب الشبكات العصبية الاصطناعية، والخوارزميات المستوحاة من الدماغ البشري، القدرة على التعلم من كميات كبيرة من البيانات. تقوم خوارزمية التعلم العميق، على غرار الطريقة التي نتعلم بها من التجارب والأمثلة، بعمل شيء واحد مرارًا وتكرارًا، وتغييره قليلاًفي كل مرة لتحسين النتيجة. من خلال القيام بذلك، فإنه يساعد أجهزة الكمبيوترفي العثور على خصائص من البيانات والتكيف مع التغييرات. يساعد التعرض المتكرر لمجموعات البيانات الآلات على فهم الاختلافات

ومناطق البيانات والوصول إلى نتيجة موثوقة.في أبسط أشكاله، يمكن اعتبار التعلم العميق وسيلة لأتمتة التحليلات التنبؤية (**predictive analytics**).

التعريف 1.1 التعلم العميق

التعلم العميق عبارة عن مجموعة من الخوارزميات التى "تتعلم من خلال الطبقات". بمعنى آخر، يتضمن التعلم من خلال الطبقات التى تمكن الخوارزمية من إنشاء تسلسل هرمى للمفاهيم المعقدة من مفاهيم أبسط.

لفهم التعلم العميق بشكل أفضل، تخيل طفلًا صغيرًا يتعلم ماهية القطة. يتعلم الطفل الصغير ما هي القطة وما هي القطة من خلال الإشارة إلى الأشياء وقول كلمة "قطة". يقول الآباء، "نعم، إنها قطة" أو "لا، إنها ليست قطة". مع استمرار الطفل الدارجفي الإشارة إلى الأشياء، فإنه يصبح أكثر وعيًا بالخصائص التي تتمتع بها جميع القطط؛ ما الذي يفعله الطفل دون أن يعرف ذلك. هذه هي الطريقة التي يخلق بها تجريدًا معقدًا (مفهوم القطة) من خلال إنشاء تسلسل هرمي يكون فيه كل مستوى من التجريد مع المعرفة المكتسبة من الطبقة السابقة للتسلسل الهرمي، لجعل هذا التجريد المعقد بسيطًا وواضحًا.

نشأة التعلم العميق؟

منذ بداية عصر الكمبيوتر، كان الباحثون يضعون نظريات حول ذكاء الآلة ويحلمون بامتلاك أجهزة كمبيوتر ذكية يمكنها تعلم وفهم الحلول للمشكلات المعقدة.في الخمسينيات والستينيات من القرن الماضي، ظهرت أولى الشبكات العصبية، بمافي ذلك خوارزمية بيرسبترون لتصنيف الصور. ومع ذلك، كانت هذه الحالات المبكرة بسيطة للغاية ولا يمكن أن تكون شائعة على نطاق واسع.

عادت الشبكات العصبية إلى الظهورفي الثمانينيات عندما طور الباحثون طرقًا لإعادة إرسال المعاملات لبناء وتدريب شبكات عصبية متعددة المستويات.

مع التطورات التي حدثتفي العقد الأول من القرن الحادي والعشرين، ظهرت تقنيات تجعل من الممكن زيادة طبقات الشبكات العصبية. أدت هذه الشبكات متعددة الطبقات إلى تسمية مجال أبحاث الذكاء الاصطناعي "التعلم العميق" لأن الخوارزميات تعالج البياناتفي طبقات متعددة للاستجابة.

في عام 2012، بدأت الشبكات العصبية العميقةفي الأداء بشكل أفضل من خوارزميات التصنيف التقليدية، بمافي ذلك خوارزميات التعلم الآلي. ترجع هذه الزيادةفي الأداء إلى حد كبير إلى زيادة أداء معالجات الكمبيوتر (GPUs) والكمية الكبيرة من البيانات المتوفرة الآن. أدت الرقمنة السريعة إلى إنتاج بيانات واسعة النطاق، وهي الأكسجين المستخدمفي تدريس نماذج التعلم

العميق. منذ ذلك الحين،في كل عام، استمر التعلم العميقفي التحسن وأصبح أفضل نهج لحل المشكلاتفي العديد من المجالات المختلفة.

يرجع هذا الارتفاع فى الشعبية واستخدام التعلم العميق إلى حد كبير إلى التقدم فى الأجهزة ومجموعات البيانات الموسومة الضخمة التى تسمح لنماذج التعلم العميق بالتعافى بسرعة بمرور الوقت.

سبب شعبية التعلم العميق؟

تتجه صناعة البرمجيات اليوم نحو الذكاء الآلي، والتعلم الآلي هو الذي مهد الطريق للآلات الذكية. ببساطة، التعلم الآلي عبارة عن مجموعة من الخوارزميات التي تحلل البيانات، وتتعلم منها، ثم تطبق ما تعلموه لاتخاذ قرارات ذكية. الشيء الذي يميز خوارزميات التعلم الآلي التقليدية هو أنها، بغض النظر عن مدى تعقيدها، لا تزال شبيهة بالآلة. بمعنى آخر، يحتاجون إلى خبراءفي هذا المجال للتعلم. بالنسبة لخبراء الذكاء الاصطناعي، هذه هي النقطة التي يعد بها التعلم العميق. وذلك لأن الشبكات العصبية العميقة تتعلم ميزات عالية المستوى من البيانات بشكل تدريجي (هرميًا) دون الحاجة إلى تدخل بشري. هذا يلغي الحاجة إلى خبراء المجال واستخراج الميزات يدويًا. اختيار السمات لمجموعة البيانات له تأثير كبير على نجاح نموذج التعلم الآلي،في حين أن استخراج السمات يدويًا سيكون عملية معقدة وتستغرق وقتًا طويلاً.

اليوم، بالإضافة إلى الشركات والمؤسسات، حتى الأشخاصفي الجوانب التكنولوجية يميلون إلى التعلم العميق، وعدد هذه الشركات والأفراد الذين يستخدمون التعلم العميق يتزايد يومًا بعد يوم. لفهم هذا السبب، يجب على المرء أن ينظر إلى الفوائد التي يمكن اكتسابها باستخدام نهج التعلم العميق. يمكن تلخيص الفوائد الرئيسية لاستخدام هذه التقنية على النحو التالي:

- **لا حاجة لهندسة الميزات:** في التعلم الآلي، تعد هندسة الميزات مهمة أساسية وهامة. هذا لأنه يحسن الدقة، وفي بعض الأحيان قد تتطلب هذه العملية معرفة المجال حول مشكلة معينة. تتمثل إحدى أكبر مزايا استخدام نهج التعلم العميق في قدرته على أداء هندسة الميزات تلقائيًا. في هذا النهج، تقوم الخوارزمية بمسح البيانات لتحديد الميزات ذات الصلة ثم دمجها لتسريع التعلم، دون إخبارها صراحة. تساعد هذه الإمكانية علماء البيانات على توفير قدر كبير من الوقت ثم تحقيق نتائج أفضل.
- **الاستخدام الاقصى للبيانات غير المهيكلة:** تظهر الأبحاث أن نسبة كبيرة من بيانات المؤسسة غير المهيكلة ، لأن معظمها في تنسيقات مختلفة مثل الصور والنصوص وما إلى ذلك. بالنسبة لمعظم خوارزميات التعلم الآلي، يعد تحليل البيانات غير المهيكلة أمرًا صعبًا. هذا هو المكان الذي يكون فيه التعلم العميق مفيدًا. لأنه يمكنك استخدام تنسيقات بيانات مختلفة لتعليم خوارزميات التعلم العميق وأيضًا اكتساب رؤى تتعلق

بالغرض من التدريب. على سبيل المثال ، يمكنك استخدام خوارزميات التعلم العميق لاكتشاف العلاقات بين تحليل الصناعة ودردشة الوسائط الاجتماعية والمزيد للتنبؤ بأسعار الأسهم المستقبلية للمؤسسة.

- **تقديم نتائج عالية الجودة:** يصاب البشر بالجوع أو التعب وأحيانًا يرتكبون أخطاء. في المقابل، ليس هذا هو الحال عندما يتعلق الأمر بالشبكات العصبية. يمكن لنموذج التعلم العميق المدرب بشكل صحيح أن ينجز آلاف المهام الروتينية والمتكررة في فترة زمنية قصيرة نسبيًا مقارنة بما يحتاجه الإنسان. بالإضافة إلى ذلك ، لن تنخفض جودة العمل أبدًا ، ما لم تحتوي بيانات التدريب على بيانات أولية لا تشير إلى مشكلة تريد حلها.
- **التعلم الانتقالي:** يحتوي التعلم العميق على العديد من النماذج المدربة مسبقًا بأوزان وتحيزات ثابتة ، وبعضها ممتاز في التنبؤ.
- **دقة عالية في النتائج:** عندما يتم تعليم التعلم العميق بكميات هائلة من البيانات، يمكن أن يكون دقيقًا للغاية مقارنة بخوارزميات التعلم الآلي التقليدية.

بالنظر إلى المزايا المذكورة أعلاه والاستفادة بشكل أكبر من نهج التعلم العميق، يمكن القول إن التأثير الكبير للتعلم العميق على التقنيات المتقدمة المختلفة مثل إنترنت الأشياء في المستقبل واضح. لقد قطع التعلم العميق شوطًا طويلاً وأصبح سريعًا تقنية حيوية يتم استخدامها باستمرار من قبل مجموعة من الشركاتفي مجموعة متنوعة من الصناعات.

ومع ذلك، تجدر الإشارة إلى أن التعلم العميق قد لا يكون أيضًا الخيار الأفضل استنادًا إلى البيانات. على سبيل المثال، إذا كانت مجموعة البيانات صغيرة، فقد تؤدى أحيانًا نماذج تعلم الآلة الخطية الأبسط إلى نتائج أكثر دقة. ومع ذلك، يجادل بعض خبراء التعلم الآلى بأن شبكة عصبية عميقة جيدة التدريب لا يزال بإمكانها العمل بشكل جيد مع كميات صغيرة من البيانات.

كيف يعمل التعلم العميق؟

تكتسب نماذج التعلم العميق القدرة على التعلم من خلال التحليل المستمر للبيانات واكتشاف الهياكل المعقدةفي البيانات. تتحقق عملية التعلم من خلال بناء نماذج حسابية تسمى الشبكات العصبية المستوحاة من بنية الدماغ.في صميم هذا التعلم يوجد نهج متكرر لتدريب الآلات لتقليد الذكاء البشري. تقوم الشبكة العصبية الاصطناعية بتنفيذ هذه الطريقة التكرارية من خلال عدة مستويات هرمية، حيث تكون قادرة على حل مفاهيم أكثر تعقيداً للمشكلة من خلال الانتقال إلى طبقات المستوى التالي. تساعد المستويات الأساسية الآلات على تعلم معلومات بسيطة. كلما انتقلت إلى كل مستوى جديد، تجمع الأجهزة المزيد من المعلومات وتدمجها مع ما تعلمتهفي

المستوى الأخير.في نهاية العملية، يجمع النظام جزءًا أخيرًا من المعلومات يمثل إدخالًا مختلطًا. تمر هذه المعلومات بعدة تسلسلات هرمية وتشبه التفكير المنطقي المعقد.

دعنا نقسمها أكثر بمساعدة مثال. ضعفي اعتبارك مساعداً صوتيًا مثل Alexa أو Siri لترى كيف يستخدم التعلم العميق لتجارب المحادثة الطبيعية.في المراحل الأولى من الشبكة العصبية، عندما يتغذى المساعد الصوتي على البيانات، فإنه يحاول تحديد الأصوات والأشياء الأخرى.في المستويات العليا، يلتقط معلومات المفردات ويضيف النتائج من المستويات السابقة.في المستويات التالية، يحلل الإعلانات (الأوامر) ويجمع كل نتائجها. بالنسبة لأعلى مستوى من الهيكل الهرمي، يتم تدريب المساعد الصوتي بدرجة كافية ليكون قادرًا على تحليل الحوار وتقديم المدخلات بناءً على تلك المدخلات.

فى التعلم العميق، لا نحتاج إلى برمجة صريحة لكل شيء. يمكنهم التعرف تلقائيًا على تمثيلات البيانات مثل الصور أو الفيديو أو النص، دون تقديم قواعد يدوية. يمكن أن تتعلم البنى شديدة المرونة الخاصة بهم مباشرة من البيانات الخام ويمكن أن تزيد من الأداء إذا تم توفير المزيد من البيانات.

عيوب وتحديات التعلم العميق

على الرغم من تزايد أهمية التعلم العميق وتطوراته، إلا أن هناك بعض الجوانب أو التحديات السلبية التي يجب معالجتها لتطوير نموذج التعلم العميق. أكبر قيود على نماذج التعلم العميق هو أنها تتعلم من خلال الملاحظة. هذا يعني أنهم يعرفون فقط ما هو موجودفي البيانات التي يقومون بتعليمها وأنهم جيدون فقطفي التعيين بين المدخلات والمخرجات. بمعنى آخر، لا يعرفون شيئًا عن سياق البيانات التي يستخدمونها.في الواقع، تشير كلمة "عميق"في التعلم العميق إلى مرجع هندسة التكنولوجيا وعدد الطبقات المخفيةفي هيكلها أكثر من كونها تشير إلى فهم عميق لما يتم القيام به.

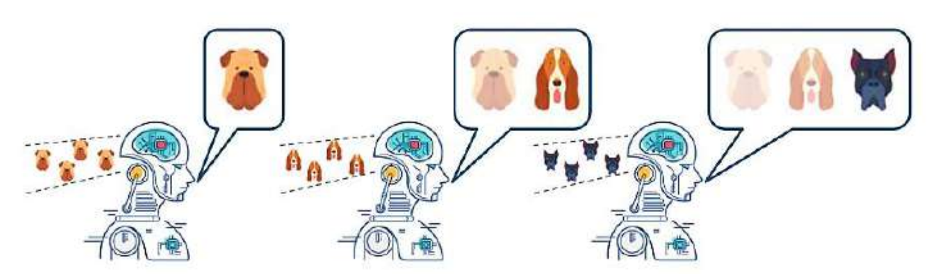
تعد نماذج التعلم العميق واحدة من أسوأ نماذج البياناتفي عالم التعلم الآلي. إنهم بحاجة إلى قدر هائل من البيانات لتحقيق الأداء المطلوب وخدمتنا بالقدر التي نتوقعها منهم. ومع ذلك، فإن الحصول على هذا القدر من البيانات ليس بالأمر السهل دائمًا. بالإضافة إلى ذلك،في حين أنه يمكننا الحصول على كميات كبيرة من البيانات حول موضوع ما، إلا أنه غالبًا لا يتم تصنيفها، لذلك لا يمكننا استخدامها لتعليم أي نوع من خوارزمية التعلم الخاضع للإشراف. باختصار، لا تتعلم هذه النماذج بطريقة قابلة للتعميم إذاكان لدى المستخدم كمية صغيرة من البيانات. يمكن أن يعمل التعلم العميق بشكل أفضل عند توفر كمية كبيرة من البيانات عالية الجودة. مع زيادة البيانات المتاحة، يزداد أداء نظام التعلم العميق.

عندما لا يتم إدخال بيانات ذات جودة فى النظام، يمكن أن يفشل نظام التعلم العميق فشلاً ذريعاً.

تعد قضية **التحيزات (biases)** أيضًا مشكلة رئيسية لنماذج التعلم العميق. إذا تم تدريب نموذج على بيانات متحيزة، فإن النموذج يعيد إنتاج تلك التحيزاتفي تنبؤاته.

على الرغم من أن نماذج التعلم العميق فعالة للغاية ويمكنها صياغة حل مناسب لمشكلة معينة بعد التدريب مع البيانات، إلا أنها غير قادرة على القيام بذلك لحل مشكلة مماثلة وتحتاج إلى إعادة التدريب. لتوضيح ذلك، ضعفي اعتبارك خوارزمية التعلم العميق التي تتعلم أن الحافلات المدرسية دائمًا ما تكون صفراء، ولكن فجأة تتحول الحافلات المدرسية إلى اللون الأزرق. ومن ثم، يجب إعادة تدريبها. على العكس من ذلك، ليس لدى الطفل البالغ من العمر خمس سنوات مشكلةفي التعرف على السيارة كحافلة مدرسية زرقاء. بالإضافة إلى ذلك، فهي أيضًا لا تعمل بشكل جيدفي المواقف التي قد تكون مختلفة قليلاً عن البيئة التي مارسوا فيها. على سبيل المثال،في DeepMind دربت Google على هزيمة 49 لعبة Atari. ومع ذلك،في كل مرة يهزم فيها النظام مباراة واحدة، يجب إعادة تدريبه لهزيمة المباراة التالية. يقودنا هذا إلى تحديد آخر للتعلم العميق، وهو أنهفي حين أن النموذج قد يكون جيداً للغايةفي تعيين المدخلات إلى المخرجات، فقد لا يكون جيداًفي فهم سياق البيانات التي يديرونها.

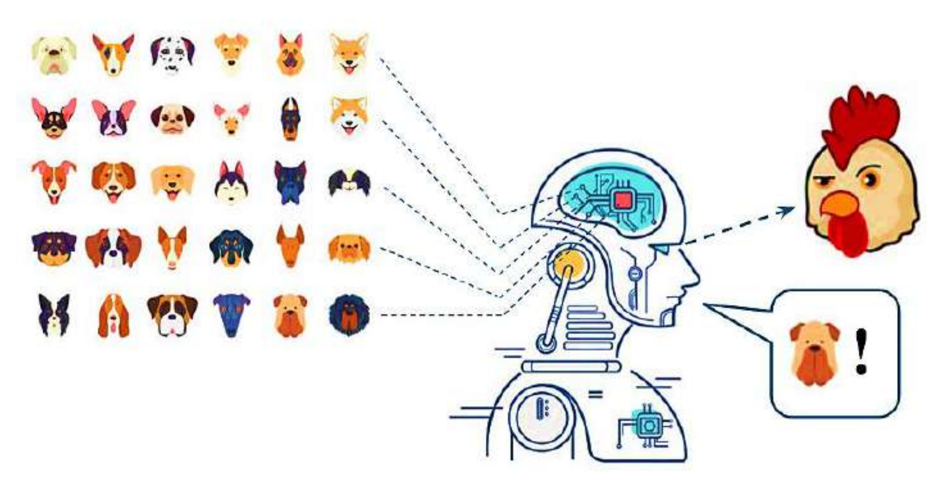
يتعلم نمط التعلم العميق أو بشكل عام خوارزميات التعلم الآلي الحالية بشكل منفصل: وفقًا لمجموعة بيانات التدريب، تعمل خوارزمية التعلم الآلي على مجموعة البيانات لإنتاج نموذج ولا تبذل أي جهد للحفاظ على المعرفة المكتسبة واستخدامهافي تفعيل التعلمفي المستقبل. على الرغم من أن نموذج التعلم المنفصل هذا كان ناجحًا للغاية، إلا أنه يتطلب عددًا كبيرًا من أمثلة التدريب وهو مناسب فقط للمهام المحددة والمحدودة جيداً. مع توفر مجموعات بيانات أكبر وخفض التكاليف الحسابية، أصبحت النماذج القادرة على حل المهام الأكبر متاحة. ومع ذلك، قد يكون تعليم نموذجفي كل مرة يحتاج فيها لتعلم مهمة جديدة أمرًا مستحيلًا. نظرًا لأن البيانات القديمة قد لا تكون متاحة، فقد لا يتم تخزين البيانات الجديدة بسبب مشكلات الخصوصية، أو قد لا يدعم تكرار تحديث النظام تدريب نموذج جديد مع تكرار جميع البيانات بشكل كافٍ. عندما تتعلم الشبكات العصبية العميقة مهامًا جديدة، فإن المعرفة الجديدة لها الأسبقية على المعرفة القديمة إذا لم يتم استخدام معايير معينة، مما يؤدي غالبًا إلى نسيان المعرفة الثانوية. يُعرف هذا **بالنسيان الكارثي (catastrophic forgetting)** (انظر الشكل 1ـ1). يحدث النسيان الكارثي عندما تكون الشبكة العصبية المدربة غير قادرة على الحفاظ على قدرتها على أداء المهام التي تعلمتها بالفعل عندما تتكيف مع المهام الجديدة.



الشكل 1ـ1. صورة النسيان الكارثي. تُنسى المعرفة التي تم تعلمها سابقًا (تختفي تدريجياً) عند تعلم فئات جديدة لم تتم رؤيتها لفترة من الوقت.

مشكلة أخرى مع الشبكات العصبية العميقة هي أنها غالبًا ما يتم تدريبها على افتراضات العالم المغلق، أي أنه من المفترض أن يكون توزيع البيانات التجريبية مشابهًا لتوزيع بيانات التدريب. ومع ذلك، عند استخدامه في العمل الواقعي، يكون هذا الافتراض غير صحيح ويؤدي إلى انخفاض كبير في أدائهم. عندما تعالج الشبكات العصبية العميقة البيانات التي لا تشبه التوزيع الذي لوحظ أثناء التدريب، ما يسمى **خارج التوزيع** (**Out-of-distribution**)، فإنها غالبًا ما تقدم تنبؤات خاطئة وتفعل ذلك بثقة كبيرة. (انظر الشكل 1ـ2). في هذه الحالات ، يرتبط إخراج الشبكة ارتباطًا مباشرًا بحل المشكلة ، أي احتمال كل فئة. ومع ذلك، يجب أن يكون مجموع تمثيلات متجه الإخراج واحدًا دائمًا. هذا يعني أنه عندما يتم عرض إدخال على الشبكة ليس جزءًا من توزيع التدريب ، فإنه لا يزال يعطي الاحتمال لأقرب فئة بحيث يصل مجموع الاحتمالات إلى واحد. أدت هذه الظاهرة إلى المشكلة المعروفة المتمثلة في الإفراط في الثقة (overconfident) في الشبكات العصبية بمحتوى لم يسبق لهم رؤيته من قبل.

على الرغم من أن هذا الانخفاض في الأداء مقبول لتطبيقات مثل التوصية بالمنتجات، إلا أن استخدام مثل هذه الأنظمة في مجالات مثل الطب والروبوتات المنزلية يعد أمرًا خطيرًا ، حيث يمكن أن يتسببوا في حوادث خطيرة. ينبغي ، إن أمكن ، تعميم نظام الذكاء الاصطناعي المثالي على عينات خارج التوزيع. لذلك ، تعد القدرة على الكشف عن التوزيعات أمرًا بالغ الأهمية للعديد من تطبيقات العالم الحقيقي وهي ضرورية لضمان موثوقية وسلامة أنظمة التعلم الآلي. في القيادة الذاتية، على سبيل المثال ، نريد أن يقوم نظام القيادة بالتحذير وتسليم السيطرة عندما يكتشف مشاهد أو أشياء غير عادية لم يراها من قبل ولا يمكنه اتخاذ قرار آمن.



الشكل 1ــ2. عندما يتم تقديم عينة جديدة خارج التوزيع المكتسب، فإن الشبكات العصبية العميقة لفئة معينة تتنبأ بثقة بالتوزيع المكتسب.

أخيرًا، يتمثل أكثر نقاط الضعف المعروفةفي الشبكات العصبيةفي افتقارها إلى الشفافية. بينما يمكن تتبع القرارات التي تتخذها النماذج المستندة إلى القواعد من خلال عبارات if and else، فإن هذا لن يكون ممكنًافي التعلم العميق. هذا النقصفي الشفافية هو ما يشار إليهفي التعلم العميق باسم **"الصندوق الأسود"**.

ببساطة، أنت لا تعرف كيف ولماذا حصلت شبكتك العصبية على ناتج معين. على سبيل المثال، عندما تقوم بإدخال صورة قطة إلى شبكة عصبية ويتنبأ بها الجهاز، فمن الصعب جدًا فهم سبب هذا التوقع. سيكون هذا السيناريو مهمًافي قرارات العمل. هل يمكنك أن تتخيل أن الرئيس التنفيذي لشركة كبيرة يتخذ قرارًا بشأن ملايين الدولارات دون أن تفهم سبب قيامه بذلك؟ فقط لأن "الكمبيوتر" يقول إنه يجب أن يفعل ذلك؟ بالمقارنة، الخوارزميات مثل أشجار القرار قابلة للتفسير بشكل كبير.

تجد خوارزميات التعلم العميق الأنماط والارتباطات من خلال البيانات التي يتم تغذيتها بها، وفي بعض الأحيان تتخذ قرارات مربكة حتى للمهندسين الذين قاموا بإنشائها. لن تكون هذه مشكلة عندما يقوم التعلم العميق بشيء ذي أهمية ثانوية. ولكن عندما يتعلق الأمر بتقرير مصير المتهم في المحكمة أو العلاج الطبي لمريض، فقد يكون ذلك أمرًا بالغ الأهمية. لأن الأخطاء يمكن أن يكون لها عواقب كثيرة. بحسب ماركوس:

"لا تزال مشكلة الشفافية دون حل ، ويريد المستخدمون فهم كيفية اتخاذ نظام معين لقرار محدد عند استخدام التعلم العميق للعمل في مجالات التشخيص الطبي والأعمال المالية."

بشكل عام ، وفقًا لأندرو آنج ، يعد التعلم العميق طريقة رائعة "لبناء مجتمع قائم على الذكاء الاصطناعي"، والتغلب على أوجه القصور هذه بمساعدة التقنيات الأخرى هو الطريق الصحيح للوصول الى الهدف.

مستقبل التعلم العميق؟

يعد التعلم العميق حاليًا أكثر تقنيات الذكاء الاصطناعي فاعلية لتطبيقات متعددة. ومع ذلك، هناك آراء مختلفة حول قدرات التعلم العميق. بينما يعتقد بعض الباحثين في التعلم العميق أنه يمكن حل جميع المشكلات عن طريق التعلم العميق، هناك العديد من العلماء الذين يشيرون إلى أوجه القصور في التعلم العميق.

عالم النفس البحثي غاري ماركوس، أحد الرواد في مجال التعلم العميق !!!، يقترح طرقًا جديدة لتحسين حلول التعلم العميق. تتضمن هذه الأساليب تقديم المنطق أو المعرفة السابقة للتعلم العميق، وتعلم المراقبة الذاتية، وشبكات الكبسولة، وما إلى ذلك. يؤكد جاري ماركوس أن تقنيات التعلم العميق كثيفة البيانات وهشة، وأن قدرتها على التعميم محدودة.

يقول عالم الكمبيوتر إيان لاكان: "لا يمكن لأي من تقنيات الذكاء الاصطناعي التي نمتلكها إنشاء تمثيلات للعالم قريبة مما نراه في الحيوانات والبشر، من خلال البنية أو التعلم". ومن ثم، فإن تقنيات الذكاء الاصطناعي الحالية مثل التعلم العميق لا تزال غير قادرة على إنشاء ذكاء اصطناعي عام يتمتع بذكاء يمكن مقارنته بالحيوانات أو البشر. ومع ذلك، يعتقد لاكان أن الذكاء الاصطناعي يمكن أن يعزز تطوير الذكاء العام على أساس التعلم العميق دون إشراف. يلبي التقدم الحديث الحاجة البشرية إلى البيانات ذات العلامات اليدوية التي تتعلم منها الآلات.

يقترح غاري ماركوس، مؤيد للنهج الهجين للتعلم العميق، خطة من أربع خطوات لمستقبل التعلم العميق:

1. **الاتصال بعالم الذكاء الاصطناعي الكلاسيكي.** لا يقترح ماركوس تحرير التعلم العميق، لكنه يجادل بأنه يجب علينا استخدام مناهج أخرى للذكاء الاصطناعي مثل المعرفة السابقة، والتفكير، والنماذج المعرفية الغنية جنبًا إلى جنب مع التعلم العميق من أجل التغيير التحويلي.
2. **بناء أطر معرفية ثرية وقواعد بيانات معرفية واسعة النطاق.** تمتلئ أنظمة التعلم العميق إلى حد كبير فقط بالارتباطات بين أشياء محددة. لذلك هم بحاجة إلى الكثير من المعرفة.
3. **أدوات للتفكير المجرد من أجل التعميم الفعال.** يجب أن نكون قادرين على الجدال حول هذه الأشياء. لنفترض أن لدينا معلومات عن الأشياء المادية وموقعها في الكون، على سبيل المثال الكوب. يحتوي الكأس على قلم رصاص. يجب أن تكون أنظمة التعلم العميق قادرة بعد ذلك على اكتشاف أن أقلام الرصاص قد تسقط إذا قمنا بعمل ثقب في قاع الكوب. يقوم البشر دائمًا بهذا النوع من الاستنتاجات، لكن أنظمة التعلم العميق الحالية، أو الذكاء الاصطناعي بشكل عام ، لا تملك هذه القدرة.
4. **آليات تمثيل واستقراء النماذج المعرفية.**

لا يزال أمام التعلم العميق طريق طويل للوصول إلى قدرات نظرائه من البشر.

التفكير الرمزي (symbolic reasoning)

يوصف تاريخ أبحاث الذكاء الاصطناعي أحيانًا بأنه تضارب بين نهجين مختلفين للتفكير الرمزي والتعلم الآلي.في العقد الأول، ساد التفكير الرمزي، لكن التعلم الآلي بدأ يتخلل التسعينيات وانتشر في جميع أنحاء المجال مع ثورة التعلم العميق. ومع ذلك، يبدو أن التفكير الرمزي هو مجرد مجموعة أخرى من الأساليب التي قد تؤدي إلى توسيع وتمكين التعلم العميق.

يقول نيكو ستروم: "تمتلك شبكات المحولات (Transformer networks) شيئًا يسمى الانتباه"(**attention**). يمكنك تعبئتها مسبقًا بالمتجهات التي تمثل الحقيقةفي قاعدة المعرفة هذه، وبعد ذلك يمكنك مطالبة الشبكة بالاهتمام بالمعرفة الصحيحة اعتمادًا على المدخلات، لذلك يمكنك محاولة هيكلة معرفة العالم من خلال الجمع بين نظام التعلم العميق.

هناك أيضًا شبكات عصبية للرسم البياني يمكنها تمثيل المعرفة حول العالم. لديك عقد وحواف بين هذه العقد تشير إلى العلاقات بينها. لذلك، على سبيل المثال، يمكنك إظهار الكياناتفي العقد ثم العلاقات بين الكيانات. يمكننا استخدام الانتباهفي جزء من الرسم البياني المعرفي ذي صلة بالسياق أو السؤال الحالي.

يبدو أنه يمكن تمثيل كل المعارففي رسم بياني واحد. ومع ذلك، فإن النقطة المهمة هي، كيف يمكن القيام بذلك بكفاءة وبشكل مناسب؟

"كانت لدى هينتون هذه الفكرة منذ وقت طويل. وقد أطلق عليها اسم **ناقل الفكر** (**thought vector**). يمكن تمثيل أي فكرة لديك بواسطة ناقل. الشيء المثير للاهتمام هو أنه يمكننا إظهار أي شيء على رسم بياني. ولكن لكي يعمل هذا بشكل جيد باستخدام نموذج التعلم العميق، يجب أن يكون لدينا شيء من ناحية أخرى يمكننا تمثيل كل شيء به، وهذا متجه، حتى نتمكن من إنشاء خريطة بين الاثنين. "

تطبيقات التعلم العميق؟

يستخدم التعلم العميق الآن كأداة قوية وشائعة لحل المشاكل البشريةفي كل مجال. يستخدم التعلم العميق لمئات المشاكل، من رؤية الكمبيوتر إلى معالجة اللغة الطبيعية.في كثير من الحالات، كان التعلم العميق أفضل من ذي قبل. يستخدم التعلم العميق على نطاق واسعفي الجامعات لدراسة الذكاء وفي الصناعة لبناء أنظمة ذكية لمساعدة البشرفي مجموعة متنوعة من المهام.في هذا القسم، سوف نلقي نظرة على بعض تطبيقات التعلم العميق التي من المؤكد أنها ستدهشك. بالطبع، هناك العديد من البرامج المختلفة وهذه القائمة ليست شاملة بأي حال من الأحوال.

المساعدين الافتراضيين

أشهر المساعدين الافتراضيين هما Siri و Alexa و Google Assistant . يوفر كل تفاعل مع هؤلاء المساعدين فرصة لهم لمعرفة المزيد عن صوتك ولغتك، مما يؤدي إلى تجربة ثانوية للتفاعل البشري. يستخدم المساعدون الافتراضيون التعلم العميق لمعرفة المزيد حول موضوعاتهم، من تفضيلات الطعام الخاصة بك إلى النقاط الساخنة أو الموسيقى المفضلة لديك. سوف يتعلمون فهم أوامرك من خلال تقييم اللغة البشرية الطبيعية لتنفيذها.

هناك ميزة أخرى تُمنح للمساعدين الافتراضيين وهي ترجمة كلامك إلى نص وتدوين الملاحظات نيابة عنك وحجز موعد. المساعدون الافتراضيون موجودونفي خدمتك حرفيًا، حيث يمكنهم التعامل مع كل شيء بدءًا من المهام وحتى الرد التلقائي على مكالماتك المحددة وتنسيق المهام بينك وبين أعضاء فريقك.

جمع الأخبار واكتشاف أخبار الاحتيال

توجد الآن طريقة لتصفية جميع الأخبار السيئة والقبيحة من اختيارك للأخبار. يعزز الاستخدام المكثف للتعلم العميقفي تجميع الأخبار الجهود المبذولة لتخصيص الأخبار وفقًا للقراء. على الرغم من أن هذا قد لا يبدو جديدًا، فقد تم تطوير مستويات جديدة من التعقيد لتحديد شخصيات القارئ لتصفية الأخبار بناءً على المعايير الجغرافية والاجتماعية والاقتصادية جنبًا إلى جنب مع التفضيلات الفردية للقارئ. من ناحية أخرى، يعد اكتشاف الاحتيال من الأصول المهمة في عالم اليوم حيث أصبح الإنترنت المصدر الأساسي لجميع المعلومات الصحيحة والكاذبة.

من الصعب جدًا اكتشاف الأخبار المزيفة لأن الروبوتات تكررها تلقائيًا على القنوات. يساعد التعلم العميقفي تطوير الفئات التي يمكنها اكتشاف الأخبار الزائفة أو المتحيزة وإزالتها من مقتطفات الأخبار الخاصة بك وتحذيرك من انتهاكات الخصوصية المحتملة.

الذكاء العاطفي

في حين أن أجهزة الكمبيوتر قد لا تكون قادرة على تكرار المشاعر البشرية، فإنها تكتسب فهمًا أفضل لمواقفنا بفضل التعلم العميق. أنماط مثل التغييراتفي النغمة، أو التجهم الخفيف، أو البحة كلها إشارات بيانات قيّمة يمكن أن تساعد الذكاء الاصطناعي واكتشاف الحالة المزاجية لدينا. يمكن استخدام مثل هذه البرامج لمساعدة الشركات على ربط البيانات العاطفية بالإعلانات، أو حتى لإعلام الأطباء بحالة المريض العاطفية.

الرعاية الصحية

لا يمكن للأطباء أن يكونوا مع مرضاهم على مدار 24 ساعةفي اليوم، ولكن الشيء الوحيد الذي نحمله معنا دائمًا هو هواتفنا. بفضل التعلم العميق، يمكن للأدوات الطبية فحص البيانات التي نأخذها وبيانات الحركة لتشخيص المشاكل الصحية المحتملة. يستخدم برنامج رؤية الكمبيوتر Robbie.AIهذه البيانات لتتبع أنماط حركة المريض للتنبؤ بالسقوط بالإضافة إلى التغيراتفي الحالة العقلية للمستخدم. أثبت التعلم العميق أيضًا أنه يشخص سرطان الجلد من خلال الصور.

يستخدم التعلم العميق أيضًا على نطاق واسعفي التجارب السريرية لإيجاد حلول للأمراض المستعصية، لكن شكوك الأطباء ونقص قواعد البيانات الشاملة لا يزالان يشكلان تحديات لاستخدام التعلم العميقفي الطب.

كشف الاحتيال

مع استمرار البنوكفي رقمنة عمليات معاملاتها، تزداد احتمالية الاحتيال الرقمي، لذلك يلعب التعلم العميق دورًا مهمًافي منع هذا النوع من الاحتيال. يمكن الكشف عن الاحتيال بسرعة من خلال تقنيات التعلم المتعمق.

خلاصه الفصل

- **الذكاء الاصطناعي هو نظام يمكنه محاكاة السلوك البشري.**
- **في أنظمة الكمبيوتر، هناك خبرة في شكل بيانات.**
- **تقوم خوارزمية التعلم العميق، على غرار الطريقة التي نتعلم بها من التجارب والأمثلة، بعمل شيء واحد مرارًا وتكرارًا ، وتغييره قليلاً في كل مرة لتحسين النتيجة.**
- **تعد نماذج التعلم العميق واحدة من أسوأ نماذج البيانات في عالم التعلم الآلي.**
- **أكثر نقاط الضعف المعروفة في الشبكات العصبية هي عدم وجود الشفافية.**

اختبار

1. **عرف الذكاء الاصطناعي والتعلم الآلي والتعلم العميق.**
2. **متى يحدث النسيان الكارثي في الشبكات العصبية؟**
3. **ما هي الميزة المهمة للتعلم العميق مقارنة بالتعلم الآلي؟**
4. **صف بعض القيود والتحديات في التعلم العميق.**

2

الأساسيات

اهداف التعليم:

- ما هي البيانات؟
- أنواع البيانات
- التعرف على الطرق المختلفة للتعلم الآلي
- مزايا وعيوب كل طريقة
- ما هو اختيار النموذج وتقييمه؟

المقدمة

التعلم العميق هو مجموعة فرعية من أساليب التعلم الآلي. وبالتالي، سيكون من المفيد مراجعة مفاهيم التعلم الآلي قبل التعلم العميق. يوضح هذا الفصل بالتفصيل المفاهيم اللازمة لفهم التعلم العميق بطريقة تقلل من المتطلبات الأساسية وتوفر معلومات كاملة حول البيانات والأدوات المطلوبة وأساسيات التعلم الآلي. بشكل عام، هذا الفصل هو الأساس لمعرفة المحتوى الكامل لهذا الكتاب.

البيانات (Data)

كلمة "البيانات" مشتقة من الكلمة اللاتينية dare، والتي تعني "شيء معين"؛ ملاحظة أو حقيقة حول موضوع ما. تأتي البياناتفي مجموعة متنوعة من الأشكال والتنسيقات، ولكن يمكن اعتبارها عمومًا تجربة عشوائية؛ تجربة لا يمكن تحديد نتيجتها مسبقًا، لكن أداؤها لا يزال خاضعًا للتحليل. غالبًا ما يتم تخزين بيانات الاختبار العشوائيفي جدول. الاصطلاح الإحصائي هو إظهار المتغيرات، التي تسمى غالبًا سمات، كأعمدة وعناصر مفردة كصفوف.

التعريف 1.2 البيانات

تشير البيانات إلى أجزاء مميزة من المعلومات التى عادة ما يتم تنسيقها وتخزينها لتناسب غرضًا محددًا. يمكن أن تأتى البيانات بأشكال عديدة، كأرقام أو نصوص على الورق، أو بتات أو بايت مخزنة فى الذاكرة الإلكترونية، أو كحقائق موجودة فى ذهن الشخص. ومع ذلك، فى علوم الكمبيوتر، تشير البيانات عادةً إلى المعلومات التى يتم إرسالها أو تخزينها إلكترونيًا وترجمتها بطريقة فعالة فى المعالجة أو المعالجة.

نحتاج إلى الكثير من البيانات لإدخالها لتعلم خوارزمية التعلم العميق. يمكن أيضًا تسمية كل نقطة بيانات بمثال او عينة. إذا نظرنا إلى مشكلة التعلم، فكل مدخل (متجه) سيكون له ناتج مرتبط (متجه). يمكنك التفكير فى هذه على أنها متغيرات مستقلة وغير تابعة.

البيانات المقروءة آليًا مقابل البيانات التي يمكن للبشر قراءتها

تعد البيانات أصلًا أساسيًا لجميع المؤسسات وتشكل الأساس للعديد من تطوير التطبيقات، كما يعتمد تطبيق الخوارزمية أيضًا على دقة البيانات ودقة المصدر. يعد عرض البيانات بالتنسيق الصحيح دون فقد القيمة الأصلية جزءًا مهمًا من نظام إدارة البيانات.

يمكن تصنيف جميع البيانات على أنها يمكن قراءتها آليًا أو يمكن قراءتها بواسطة الإنسان أو كليهما. تستخدم البيانات التي يمكن قراءتها بواسطة الإنسان تنسيقات لغة طبيعية (مثل ملف نصي يحتوي على رمز ASCII أو مستند PDF)، بينما تستخدم البيانات التي يمكن قراءتها آليًا

لغات كمبيوتر رسمية منظمة ليتم قراءتها بواسطة أنظمة أو برامج الكمبيوتر. يمكن للآلات والبشر قراءة بعض البيانات، مثل CSV أو HTML أو JSON.

القالب المقروء آليًا مصمم للأجهزة والآلات. يعد فهم هذا التنسيق معقدًا بالنسبة للبشر، وهناك حاجة إلى أدوات متخصصة لقراءة محتوى البيانات التي يمكن للآلة قراءتها. يمكن استخراج البيانات المقدمة بتنسيق يمكن قراءته آليًا واستخدامها لمزيد من المعالجة والتحليل دون تدخل بشري.

يمكن للبشر فهم البيانات التي يمكن قراءتها وتفسيرها. لا يتطلب تفسير البيانات معدات أو أجهزة متخصصة. هذه اللغة لها لغة طبيعية (على سبيل المثال، العربية والإنجليزية والفرنسية وما إلى ذلك) وتعرض بيانات غير منظمة. مثال على قالب يمكن قراءته من قبل الإنسان هو مستند PDF. على الرغم من أن PDF هو وسيط رقمي، إلا أن شاشات العرض لا تتطلب أي معدات أو أجهزة كمبيوتر خاصة لتفسيرها. بالإضافة إلى ذلك، عادةً ما تكون المعلومات الواردةفي مستند PDF مخصصة للبشر وليس للآلات.

البيانات في التكنولوجيا

كانت البياناتفي طليعة العديد من المحادثات الرئيسية حول التكنولوجيا. يتم تفسير الابتكارات الجديدة باستمرار على البيانات وكيفية استخدامها وتحليلها. نتيجة لذلك، تشتمل اللغة العامة للتكنولوجيا على عدد من التعبيرات الجديدة والقديمة:

- **البيانات الضخمة (Big data)**: كمية هائلة من البيانات المهيكلة وغير المهيكلة التي يتم إنشاؤها والحصول عليها بسرعة من مصادر مختلفة وتزيد من البصيرة واتخاذ القرار.
- **تحليلات البيانات الضخمة (Big data analytics)**: عملية جمع مجموعات البيانات الضخمة وتنظيمها والجمع بينها لاكتشاف الأنماط أو المعلومات المفيدة الأخرى.
- **تكامل البيانات (Data integrity)**: صلاحية البيانات التي يمكن اختراقها بعدة طرق، بما في ذلك الخطأ البشري أو أخطاء الإرسال.
- **البيانات الوصفية (Metadata)**: معلومات موجزة حول مجموعة البيانات.
- **البيانات الخام (Raw data)**: المعلومات التي تم جمعها ولكن لم يتم تنسيقها أو تحليلها.
- **البيانات المهيكلة (Structured data)**: أي بيانات في حقل ثابت في سجل أو ملف ، بما في ذلك البيانات الموجودة في قواعد البيانات وجداول البيانات العلائقية. بعبارة ابسط، تتكون البيانات المنظمة من أنواع محددة من البيانات ذات أنماط تجعلها سهلة البحث.
- **البيانات غير المهيكلة (Unstructured data)**: تختلف المعلومات الموجودة في قاعدة بيانات الصفوف والأعمدة التقليدية عن البيانات غير المهيكلة. بعبارة آخرى، تحتوي

البيانات غير المهيكلة على بيانات لا يمكن عادةً البحث عنها بسهولة، بما في ذلك تنسيقات مثل الصوت والفيديو ومنشورات الوسائط الاجتماعية.

انواع البيانات

تأتي البيانات بتنسيقات وأنواع مختلفة. يوفر فهم خصائص كل سمة أو جانب معلومات حول نوع العملية التي يمكن إجراؤها على تلك السمة. على سبيل المثال، يمكن التعبير عن درجة الحرارة في بيانات الطقس بأحد التنسيقات التالية:

1. **عدد درجات مئوية (25 درجة مئوية) ، فهرنهايت أو مقياس كلفن.**
2. **وضع التسميات على أساس الطقس الحار أو البارد أو المعتدل.**
3. **عدد أيام السنة التي تقل عن الصفر درجة مئوية (20 يومًا في السنة تحت الصفر)**

تشير كل هذه الخصائص إلى درجة حرارة المنطقة، لكن لكل منها نوع مختلف من البيانات.

رقمي (Numeric) او مستمر (Continuous)

درجة الحرارة المعبر عنها بالدرجات المئوية أو الفهرنهايت هي درجة حرارة رقمية ومستمرة، لأنه يمكن التعبير عنها بالأرقام واتخاذها كقيمة بين أرقام لا نهائية. الرقم الصحيح هو شكل خاص من البيانات الرقمية التي لا تحتوي على كسور عشريةفي القيمة أو بشكل أكثر دقة، لا يحتوي على قيم غير محدودة بين الأرقام المتتالية. عادة ما تشير إلى عدد العناصر، وعدد الأيام التي تقل فيها درجات الحرارة عن 0 درجة مئوية، وعدد الطلبات، وعدد الأطفالفي الأسرة، وما إلى ذلك. إذا تم تحديد نقطة الصفر، تصبح البيانات الرقمية نسبة أو نوع بيانات حقيقي. تشمل الأمثلة درجة حرارة كلفن ورصيد الحساب المصرفي والدخل.

فئوي (Categorical) او اسمي (Nominal)

يتم تعريف بيانات الفئوية أو الاسمية على أنها بيانات تُستخدم لتسمية المتغيرات بدون أي كمية صغيرة. عادة لا يوجد ترتيب جوهري للبيانات الاسمية. على سبيل المثال، يمكن اعتبار لون الهاتف الذكي نوع بيانات اسمي. لأننا لا نستطيع مقارنة لون واحد مع الألوان الأخرى. بمعنى آخر، لا يمكن القول إن "الأحمر" أكبر من "الأزرق". كمثال آخر، لون العين هو متغير اسمي له عدة فئات (أزرق، أخضر، بني) ولا توجد طريقة لفرز هذه الفئات من الأعلى إلى الأدنى (الفئوية) أو (الاسمية).

مجموعة البيانات Dataset

هي مجموعة من البيانات التي يتم تقديمها عادةً كجدول. يمثل كل عمود متغيرًا محددًا (سمة). يتوافق كل صف مع عضو معين في مجموعة البيانات ويعين قيمًا لكل من المتغيرات. تُعرف كل قيمة بالبيانات.

التعريف 1.2 مجموعة البيانات

يمكن غالبًا اعتبار مجموعة البيانات على أنها مجموعة بيانات لها نفس الخصائص. تتضمن الأحرف الأخرى لكائن البيانات: سجل، أو نقطة، أو متجه، أو نمط، أو حدث، أو عنصر، أو مثيل، أو عينه، أو مشاهدة، أو كيان.

التعريف 1.2 خصائص البيانات

الميزة هى سمة بيانات. يمكن اعتبار الميزة كمتغير توضيحى. قد تكون هذه الميزة رقمية (ارتفاع الشجرة 3 م) أو قد تكون وصفية (لون العين الزرقاء). فى كثير من الأحيان، إذا كانت وصفية، فستحتاج إلى إعطائها علامة رقمية لإجراء العمليات الحسابية.

غالبًا ما يتم تمثيل كل نقطة بيانات بواسطة متجه سمة؛ كل إدخال فى المتجه يمثل سمة.

طرق التعلم الآلي

هناك نوعان رئيسيان من طرق التعلم الآلي: بإشراف وبدون اشراف. الفرق الرئيسي بين المنهجين هو أن التعلم تحت الإشراف يكتسب التعلم باستخدام حقيقة، أو بعبارة أخرى، لدينا معرفة مسبقة بمخرجات عيناتنا. لذلك، فإن الغرض من التعلم الخاضع للإشراف هو تعليم الدالة، من خلال النظرفي عينة من البيانات والمخرجات المعنية، أفضل تقدير للعلاقة بين المدخلات والمخرجات الظاهرةفي البيانات. عادة ما يتم التعلم الخاضع للإشرافي مجال التصنيف، عندما نريد تعيين المدخلات إلى تسميات الإخراج، أو الانحدار، عندما نريد تعيين الإدخال إلى إخراج مستمر.

من ناحية أخرى،في التعلم غير الخاضع للإشراف، لا توجد مخرجات معنونة، وبالتالي فإن الهدف هو استنتاج البنية الطبيعية لمجموعة من نقاط البيانات واكتشاف الأنماط دون أي توجيه. المهام الأكثر شيوعًافي التعلم غير الخاضع للإشراف هي التجميع (Clustering) والأبعاد.في هذه الحالات، نريد معرفة البنية الجوهرية لبياناتنا دون استخدام العلامات المتوفرة.

فى نموذج التعلم الخاضع للإشراف، تتعلم الخوارزمية على مجموعة بيانات مصنفة ولديها مفتاح استجابة يمكن للخوارزمية استخدامه لتقييم دقتها فى بيانات التدريب. فى المقابل، يستخدم النموذج غير الخاضع للإشراف بيانات غير مسماة، وتحاول

الخوارزمية فهم البنية الجوهرية للبيانات عن طريق استخراج الميزات والأنماط من تلقاء نفسها.

التعلم بإشراف

يعد التعلم الخاضع للإشراف أحد أكثر فروع التعلم الآلي استخدامًا، والذي يستخدم بيانات التدريب المصنفة لمساعدة النماذج على إجراء تنبؤات دقيقة. تعمل بيانات التدريب هنا كمشرف ومعلم للآلات، ومن هنا جاء الاسم. يعتمد التعلم الخاضع للإشراف على توليد مخرجات من التجارب السابقة (البيانات المسماة).في التعلم الخاضع للإشراف، يتم تعيين متغير الإدخال (x) إلى متغير الإخراج (y) باستخدام دالة التعيين التي تم تعلمها بواسطة نموذج التعلم الآلي.

$$y = f(x)$$

هنا يقوم بإنشاء نموذج دالي يربط المتغيرين بالهدف النهائي للتنبؤ بالتسمية الصحيحة لبيانات الإدخال.

تحتوى خوارزمية التعلم الخاضع للإشراف دائمًا على هدف أو متغير نتيجة (أو متغير تابع) يتم تحديده من خلال مجموعة من المتنبئين المقترحة (المتغيرات المستقلة). تستخدم الخوارزمية هذه المجموعة من المتغيرات لإنشاء دالة تقوم بتعيين المدخلات إلى مخرجات عشوائية. تتكرر عملية التدريب هذه حتى يحقق النموذج مستوى عالٍ من الدقة.

تعنى مجموعة البيانات المصنفة أنه يتم تمييز كل مثيل فى مجموعة بيانات التدريب بالإجابة التى يجب أن توفرها الخوارزمية. لذلك، فإن مجموعة البيانات ذات العلامات لصور الزهور تخبر النموذج بالصور المرتبطة بالورود وزهور البابونج والنرجس البرى. عندما يتم عرض صورة جديدة على النموذج، يقوم النموذج بمقارنتها مع عينات التدريب للتنبؤ بالتسمية الصحيحة.

التصنيف (Classification)

التصنيف هو عملية تعلم بأشراف، مما يعني أن خوارزمية التعلم تحاول إيجاد اتصال بين البيانات والعلامات بناءً على بيانات التدريب الموسومة مسبقًا.

فى التصنيف، يتم تحديد الفئات مسبقًا وغالبًا ما يشار إليها على أنها أهداف أو علامات أو فئات.

تُستخدم البيانات المصنفة لتعليم التصنيف بحيث تعمل بشكل جيد على بيانات الإدخال الجديدة ويمكنها التنبؤ بالفئة الصحيحة لذلك العينة. بمعنى آخر، الهدف هو العثور على تقريب جيد لـ $f(x)$ بحيث يمكنه عمل تنبؤات لبيانات لم تظهر فى عملية التدريب وتحديد الفئات التى تنتمى إليها العينة الجديد.

يمكن تقسيم مسائل التصنيف إلى منظورين مختلفين. من وجهة نظر عدد التسميات، والتي يمكن تقسيمها إلى فئتين: التصنيف احادي العلامة **(Single-Label Classification)** والتصنيف متعدد العلامات **(Multi-Label Classification)**، ومن وجهة نظر عدد الفئات، يمكن تقسيمها إلى فئتين: التصنيف الثنائي **(Binary Classification)** وتصنيف متعدد الفئات **(Multi-Class Classification)**.

التصنيف الثنائي، حيث يتم تخصيص كل حالة لواحد فقط من الفئتين المحددتين مسبقًا، هو أبسط أنواع التصنيف. يمتد التصنيف الثنائي إلى تصنيف متعدد الطبقات من خلال تحديد المزيد من الفئات.

التصنيف احادي العلامة

في تصنيف بيانات احادي العلامة، يمكن ربط كل حالة بعلامة واحدة فقط، وتتنبأ خوارزمية التصنيففي مرحلة التدريب بعلامة واحدة فقط لكل حالة جديدة. بشكل عام، يمكن تقسيم مشاكل التصنيف ذات العلامات المفردة إلى مجموعتين رئيسيتين: المشكلات الثنائية ومتعددة الفئات.

التصنيف الثنائي

مشكلة التصنيف الثنائي هي أبسط حالة لمشكلة التصنيف حيث تقتصر مجموعة الفئات على اثنين فقط.في هذا الصدد، فإننا نميز بين فئة إيجابية وفئة سلبية. مثال بسيط على مشكلة التصنيف الثنائي هو عندما ترى امرأة طبيبًا لتكتشف أنها حامل. قد تكون نتيجة الاختبار إيجابية أو سلبية.

التصنيف الثنائى هو فى الأساس نوع من التنبؤ الذى يتعامل مع أى مجموعة من مجموعتى الفئات تنتمى إليها العينة.

التصنيف متعدد الفئات

التصنيف متعدد الفئات أو التصنيف المتعدد هو تصنيف العناصر إلى فئات مختلفة. على عكس التصنيف الثنائي، الذي يقتصر على فئتين فقط، فإنه لا يوجد حد لعدد الفئات ويمكنه تصنيف أكثر من فئتين. على سبيل المثال، يعد تصنيف الأخبار إلى فئات مختلفة، وتصنيف الكتب حسب الموضوع، وتصنيف الحيوانات المختلفةفي صورة واحدة أمثلة على التصنيف متعدد الفئات.

التصنيف متعدد العلامات

في مشاكل التصنيف التقليدية، ترتبط كل عينة بعلامة فئة. ومع ذلك،في العديد من سيناريوهات العالم الحقيقي، قد يتم إقران عينة بعلامات متعددة. على سبيل المثال،في فئة الأخبار، يرتبط جزء

من الأخبار حول إطلاق Apple لجهاز iPhone الجديد بكل من العلامة التجارية وعلامة التكنولوجيا. بمعنى آخر، ترتبط كل عينة بمجموعة من العلامات بدلاً من واحدة فقط. التعلم متعدد العلامات هو سياق تعلم آلي يشير إلى التعلم من البيانات متعددة العلامات التي ترتبط فيها كل عينة بالعديد من العلامات المحتملة.

يعد التصنيف متعدد العلامات أحد أهم القضايافي معالجة اللغة الطبيعية، وخاصةفي تصنيف النص. بالإضافة إلى ذلك، يتم استخدامهفي العديد من قضايا العالم الحقيقي مثل استرجاع المعلومات وتشخيص الأمراض والمعلوماتية الحيوية. يكمن الاختلاف بين التصنيف متعدد العلامات والتصنيف متعدد العلاماتفي عدد العلامات التي يمكن تخصيصها لعينة.

التصنيف متعدد العلامات هو شكل معمم لتصنيف العلامة الواحدة، حيث يمكن ربط كل حالة بمجموعة من العلامات بدلاً من العلامة.

التوقع Regression

يتمثل الاختلاف الرئيسي بين نماذج التوقع والتصنيففي استخدام خوارزميات الانحدار للتنبؤ بالقيم المستمرة (درجات الامتحان)، بينما تتنبأ خوارزميات التصنيف بقيم منفصلة (ذكر / أنثى، صواب / خطأ). الانحدار هو عملية إحصائية تجد علاقة ذات دلالة إحصائية بين المتغيرات التابعة والمستقلة. كخوارزمية، فإنه يتنبأ بعدد مستمر. على سبيل المثال، يمكنك استخدام خوارزمية الانحدار لتحديد درجات اختبار الطلاب بناءً على عدد الساعات التي يدرسونهافي الأسبوع.في هذه الحالة، تصبح الساعات المدروسة متغيرًا مستقلاً والنتيجة النهائية لاختبار الطالب هي متغير تابع. يمكنك رسم خط الأنسب **(Fitting Line)** من خلال نقاط بيانات مختلفة لإظهار تنبؤات النموذج عند إدخال مدخلات جديدة. يمكن استخدام نفس الخط للتنبؤ بدرجات الامتحان بناءً على أداء طالب آخر.

كمثال آخر، افترض أننا نريد أن يكون لدينا نظام يمكنه التنبؤ بسعر السيارة المستعملة. المدخلات وخصائص السيارة مثل العلامة التجارية والسنة والمسافة المقطوعة وغيرها من المعلومات التي نعتقد أنها تؤثر على قيمة السيارة والمخرجات هي سعر السيارة. أو فكرفي التنقل مع روبوت متحرك (سيارة ذاتية القيادة)؛ الإخراج هو الزاوية التي يجب أن تدور فيها عجلة القيادةفي كل مرة للتقدم دون الاصطدام بالعقبات والانحرافات، ويتم توفير المدخلات بواسطة أجهزة استشعارفي السيارة مثل كاميرات الفيديو ونظام تحديد المواقع العالمي (GPS) وما إلى ذلك.

مزايا وعيوب التعلم تحت الإشراف

المزايا

- التعلم تحت الإشراف هو عملية بسيطة يمكنك فهمها.

- بعد الانتهاء من عملية التدريب، لن تحتاج إلى الاحتفاظ ببيانات التدريب في الذاكرة.
- النتيجة أكثر دقة من طريقة التعلم بدون اشراف.
- نظراً لوجود بيانات مصنفة، يمكنك بسهولة اختبار النموذج وتصحيحه.

العيوب

- التعلم الخاضع للإشراف محدود في كثير من النواحي بحيث لا يمكنه أداء بعض مهام التعلم الآلي المعقدة.
- إذا قدمنا مدخلات ليست من أي من فئات بيانات التدريب، فقد يكون الناتج تسمية فئة خاطئة. على سبيل المثال، افترض أنك تقوم بتدريس مصنف الصور باستخدام بيانات القطط والكلاب. ثم إذا أعطيت صورة زرافة، فقد يكون الناتج قطة أو كلباً، وهذا ليس صحيحاً.
- يستغرق التدريب الكثير من الوقت الحسابي.
- يعد جمع البيانات وتصنيفها أمراً مكلفاً ويستغرق وقتاً طويلاً.

التعلم بدون اشراف

يحدث التعلم بدون اشراف في التعلم الآلي عندما لا يكون هناك تسمية للبيانات أو تصنيف لها. تتمثل المهمة في فرز المعلومات غير المجمعة بناءً على بعض أوجه التشابه والاختلاف دون أي توجيه. بمعنى آخر، من المتوقع أن تعثر الآلة على أنماط وهياكل مخفية في البيانات غير المسماة بمفردها. هذا هو السبب في أنه يطلق عليه بدون اشراف. لأنه لا يوجد دليل لتعليم السيارة ما هو الصواب وما هو الخطأ. في هذا النهج، لا تعرف الآلة ما الذي تبحث عنه، ولكن يمكنها فرز البيانات بشكل مستقل والعثور على أنماط مقنعة.

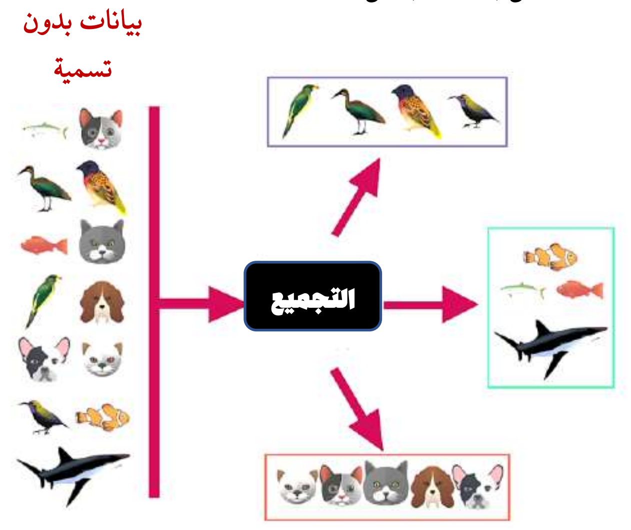
مجموعات التعلم غير الخاضعة للإشراف معلومات غير منظمة بناءً على أوجه التشابه والاختلاف، حتى لو لم يتم توفير فئات.

تتمثل إحدى الميزات المهمة لهذه النماذج في أنه بينما يمكن للنموذج اقتراح طرق مختلفة لتجميع بياناتك أو طلبها، فإن الأمر متروك لك لإجراء المزيد من البحث حول هذه النماذج للكشف عن شيء مفيد.

يعد التعلم غير الخاضع للإشراف مفيداً جداً في التحليل الاستكشافي، حيث يمكنه اكتشاف البنية المتأصلة للبيانات تلقائيًا. على سبيل المثال، إذا حاول أحد المحللين تقسيم المستهلكين، فستكون طرق التجميع غير الخاضعة للإشراف نقطة انطلاق جيدة لتحليلهم.

التجميع Clustering

التجميع (الكلاسترينك) هو عملية تعيين عينات البيانات لعدد معين من المجموعات (الشكل 2ـ1) بحيث يكون لنقاط البيانات التي تنتمي إلى المجموعة (العنقود) خصائص متشابهة. بعبارات أبسط، لا تعد المجموعات أكثر من تجميع نقاط البيانات بحيث تكون المسافة بين نقاط البيانات داخل المجموعاتفي حدها الأدنى. الهدف من تحليل المجموعة (بشكل مثالي) هو العثور على مجموعات تكون فيها العينات داخل كل مجموعة متشابهة تمامًا،في حين أن كل مجموعة مختلفة تمامًا عن بعضها البعض.



شكل 2ـ1. التجميع.

نظرًا لأن التجميع يتم بواسطة خوارزمية، فمن المحتمل أنك ستكون قادرًا على اكتشاف الارتباطات غير المعروفة سابقًا فى البيانات التى يمكن أن تساعدك فى مواجهة تحدى الأعمال من منظور جديد.

تقليل الأبعاد

يشير إلى عملية تقليل عدد السماتفي مجموعة البيانات، مع الاحتفاظ بالتغييرات على مجموعة البيانات الأصلية قدر الإمكان. تعمل عملية تقليل الأبعاد بشكل أساسي على تحويل البيانات من مساحة ميزة عالية الأبعاد إلى مساحة ميزة ذات أبعاد أصغر.في الوقت نفسه، من المهم ألا تضيع الميزات المفيدة للبيانات أثناء التحويل.

يعد تقليل الابعاد خطوة معالجة مسبقة، هذا يعنى أنه قبل تدريب النموذج، نقوم بتقليل الأبعاد.

مقارنة التعلم بالإشراف مع التعلم بدون إشراف

لنفترض أننا نريد تعليم طفل لغة جديدة، على سبيل المثال الإنجليزية. إذا فعلنا ذلك وفقًا لمبدأ التعلم الخاضع للإشراف، فإننا ببساطة نمنحه قاموسًا يحتوي على كلمات إنجليزية وترجمة إلى لغته الأم، على سبيل المثال العربية. سيكون من السهل نسبيًا أن يبدأ الطفل التعلم، وربما يمكنه التقدم بسرعة كبيرة عن طريق حفظ الترجمات. ومع ذلك، سيجد صعوبةفي قراءة وفهم النصوص باللغة الإنجليزية، لأنه تعلم فقط ترجمات الكلمات العربية ـ الإنجليزية وليس البنية النحوية للجمل الإنجليزية.

وفقًا لمبدأ التعلم غير الخاضع للإشراف، يبدو السيناريو مختلفًا تمامًا. على سبيل المثال، نمنح الطفل خمسة كتب باللغة الإنجليزية ويتعلم. طبعا هذا أكثر تعقيدا!! على سبيل المثال، بمساعدة "البيانات"، يمكن للطفل أن يدرك أن كلمة "أنا" متكررة نسبيًافي النص، وفي كثير من الحالات حتىفي بداية الجملة، ويستخلص منها استنتاجات.

يوضح هذا المثال الفرق بين التعلم الخاضع للإشراف والتعلم غير الخاضع للإشراف. التعلم الخاضع للإشراف هوفي كثير من الحالات خوارزمية أبسط. ومع ذلك، فإن النموذج يتعلم فقط السياقات المضمنة بشكل صريحفي مجموعة بيانات التدريب ويتم تقديمها كمدخلات للنموذج. على سبيل المثال، يمكن للطفل الذي يتعلم اللغة الإنجليزية أن يترجم الكلمات العربية إلى الإنجليزية جيداً، لكنه لم يتعلم قراءة وفهم النصوص الإنجليزية.

من ناحية أخرى، يعد التعلم غير الخاضع للإشراف مهمة أكثر تعقيداً لأنه يجب أن يحدد الهياكل ويتعلمها بشكل مستقل. نتيجة لذلك، يستغرق الأمر مزيداً من الوقت والجهد للممارسة. الميزة، مع ذلك، هي أن النموذج المدرب يحدد أيضا المجالات التي تم تغطيتها بشكل صريح. من المرجح أن يكون الطفل الذي تعلم اللغة الإنجليزية بمساعدة خمس روايات إنجليزية قادرًا على قراءة النصوص الإنجليزية وترجمة الكلمات الفردية إلى العربية وفهم قواعد اللغة الإنجليزية أيضًا.

لماذا التعلم بدون اشراف؟

التعلم الخاضع للإشراف فعال للغايةفي تحسين أداء المهام باستخدام مجموعات البيانات ذات العلامات المتعددة. على سبيل المثال، ضعفي اعتبارك مجموعة بيانات كبيرة جداً من صور الكائنات التي تم وضع علامة على كل صورة. إذا كانت مجموعة البيانات كبيرة بما يكفي، وإذا قمنا بتدريبها جيداً بما يكفي باستخدام خوارزميات التعلم الآلي الصحيحة وباستخدام جهاز كمبيوتر قوي، فيمكننا إنشاء نموذج تصنيف للصور قائم على التعلم يتم الإشراف عليه جيداً.

نظرًا لأنه يتم تدريب الخوارزمية عن طريق الاشراف على البيانات، يمكنها قياس أدائها (عن طريق دالة الخطأ) من خلال مقارنة علامة الصورة المتوقعة مع علامة الصورة الفعلية الموجودة لدينافي مجموعة البيانات. تحاول الخوارزمية صراحة تقليل دالة الخطأ او التكلفة هذه؛ بحيث

يكون الخطأفي الصور التي لم يتم رؤيتها من قبل (مجموعة الاختبار) أقل ما يمكن. هذا هو السببفي أن العلامات قوية جداً، فهي تساعدفي توجيه الخوارزمية من خلال توفير مقياس للخطأ. تستخدم الخوارزمية مقياس الخطأ لتحسين أدائها بمرور الوقت. بدون هذه العلامات، لا تعرف الخوارزمية مدى نجاحهافي تصنيف الصور بشكل صحيح. ومع ذلك،في بعض الأحيان تكون تكلفة وضع العلامات يدويًا على مجموعة بيانات عالية جدًا.

بالإضافة إلى ذلك، على الرغم من قوة نماذج التعلم الخاضعة للإشراف، إلا أنها محدودةفي تعميم المعرفة خارج نطاق المعرفة التي يتم تدريبهم عليها. نظرًا لأن معظم بيانات العالم غير معلمة، فإن قدرة الذكاء الاصطناعي على توسيع وظائفه لتشمل أمثلة غير مرئية من قبل محدودة باستخدام التعلم الخاضع للإشراف. بمعنى آخر، يعد التعلم الخاضع للإشراف أمرًا رائعًا لحل مشاكل **الذكاء الاصطناعي الضيق** (**Narrow AI**)، ولكنه ليس رائعًا لحل مشكلات الذكاء الاصطناعي القوية.

على النقيض من ذلك، فإن التعلم غير الخاضع للإشراف مناسب جداً للمشكلات التي تكون فيها الأنماط غير معروفة أو تتغير باستمرار أو ليس لدينا مجموعات بيانات ذات علامات كافية لها. يعمل التعلم غير الخاضع للإشراف، بدلاً من الاسترشاد بالعلامات، من خلال تعلم البنية الأساسية للبيانات التي يتم التدريب عليها. يقوم التعلم غير الخاضع للإشراف بهذا من خلال محاولة تمثيل البيانات التي يتم تدريسها عليها باستخدام مجموعة من المعاملات. من خلال القيام بهذا **التعلم التمثيلي** (**representation learning**)، يمكن للتعلم غير الخاضع للإشراف تحديد أنماط مميزةفي مجموعة البيانات.

التعريف 1.2 التعلم التمثيلي

التعلم هو تمثيل لمجموعة فرعية من التعلم الآلي الذي يهدف إلى الحصول على ميزات جيدة ومفيدة للبيانات تلقائيًا، دون إشراك مهندس ميزة. في هذا النهج، يأخذ الجهاز البيانات الأولية كمدخلات ويكتشف تلقائيًا التمثيلات اللازمة لتحديد الميزة، ثم يتعلم تلقائيًا الميزات الجديدة ويطبقها. بمعنى آخر، الهدف من التعلم التمثيلي هو إيجاد تحويل يقوم بتعيين البيانات الأولية إلى تمثيل يكون أكثر ملاءمة لمهمة تعلم الآلة (مثل التصنيف). نظرًا لأنه يمكن تفسير هذه الطريقة على أنها تعلم ميزات مفيدة، فإنها تسمى أيضًا تعلم الميزات.

في مثال مجموعة بيانات الصورة (هذه المرة غير معلمة)، قد يكون التعلم غير الخاضع للإشراف قادرًا على تحديد الصور وتجميعها بناءً على مدى تشابهها مع بعضها البعض وكيف تختلف عن الآخرين. على سبيل المثال، يتم تجميع جميع الصور التي تشبه الكراسي معًا، ويتم تجميع كل الصور التي تبدو مثل القطط معًا. بالطبع، لا يمكن للتعلم الموجه ذاتيًا تصنيف هذه المجموعات على أنها "كراسي" أو "قطط". ومع ذلك، الآن بعد أن تم تجميع الصور المتشابهة معًا، أصبح لدى البشر مهمة أبسط بكثير تتمثل في وضع العلامات. بدلاً من وضع علامات على

ملايين الصور يدويًا، يمكن للبشر وضع علامة يدويًا على جميع المجموعات المنفصلة وتطبيق هذه العلامات على جميع أعضاء كل مجموعة.

وبالتالي، فإن التعلم غير الخاضع للإشراف يجعل المشكلات التي كانت غير قابلة للحل في السابق أكثر قابلية للحل وأكثر مرونةفي العثور على الأنماط المخفية، سواءفي البيانات السابقة المتاحة للتدريب أوفي البيانات المستقبلية. حتى لو كان التعلم غير الخاضع للإشراف أقل كفاءة في حل مشكلات معينة (مشاكل محدودة للذكاء الاصطناعي) من التعلم الخاضع للإشراف، فمن الأفضل التعامل مع المشكلات الأكثر انفتاحًا من نوع أقوى من الذكاء الاصطناعي وتعميم هذه المعرفة. والأهم من ذلك، أن التعلم غير الخاضع للإشراف يمكن أن يحل العديد من المشكلات الشائعة التي يواجهها علماء البيانات عند بناء حلول التعلم الآلي.

مزايا وعيوب التعلم بدون اشراف

المزايا

- يمكنه رؤية ما لا يستطيع العقل البشري تخيله.
- الحصول على البيانات غير المعلمة أسهل نسبيًا.

العيوب

- إنها تكلف أكثر لأنها قد تتطلب تدخلاً بشريًا لفهم الأنماط وربطها بالمعرفة الميدانية.
- لا يمكن دائمًا التحقق من فائدة النتائج، لأنه لا توجد تسمية مخرجات أو معايير لتأكيد فائدتها.
- النتائج غالبا ما تكون أقل دقة.

التعليم المعزز

يتجذر التعلم المعززفي سيكولوجية تعلم الحيوانات وهو يدور حول تعلم السلوك الأمثل في بيئة للحصول على أقصى قدر من المكافأة. يتم تعلم هذا السلوك الأمثل من خلال التفاعل مع البيئة وملاحظات كيفية تفاعلها. يكافأ المتعلم (الوكيل) على الأعمال الصالحة ويعاقب على الإثم.

في حالة عدم وجود مراقب، يسعى المتعلم إلى سياسة فعالة لحل مهمة صنع القرار. تحدد مثل هذه السياسة كيفية تصرف الوكيل في أي موقف قد يواجهه من أجل تكبير (أو تقليل إجمالي المكافأة) المتوقعة عن طريق التجربة والخطأ في التفاعل مع بيئة ديناميكية. يعد التعلم المعزز خوارزمية قوية للغاية لأنه يمكن أن يتعلم الإجراءات التي تؤدي إلى النجاح النهائي في بيئة غير مرئية دون مساعدة مراقب..

اختيار النموذج وتقييمه

يمكن أن يكون لاختيار نموذج للتعلم الآلي معان مختلفة. قد نكون مهتمين باختيار أفضل **المعاملات الفائقة (Hyper Parameters)** لطريقة التعلم الآلي. المعاملات الفائقة هي معاملات طريقة التعلم التي يجب أن نحددها مسبقًا، أي قبل ملاءمة النموذج.في المقابل، معاملات النموذج هي معاملات تم إنشاؤها كنتيجة للملاءمة. على سبيل المثال،في نموذج الشبكة العصبية، يعد عدد الخلايا العصبية للطبقة المخفية وعدد الطبقات المخفية معاملات فائقة يجب تحديدها قبل التركيب،في حين أن أوزان النموذج هي معاملات النموذج. يمكن أن يكون العثور على المعاملات الفائقة المناسبة للنموذج أمرًا بالغ الأهمية لنموذج الأداء.

في أوقات أخرى، قد نرغب في تحديد أفضل طريقة تعلم من مجموعة من أساليب التعلم الآلي المؤهلة (الخوارزميات).

التعريف 2.2 اختيار النموذج (Model Selection)

اختيار نموذج تقنى لاختيار النموذج الأفضل بعد تقييم كل نموذج بناءً على المعايير المطلوبة.

قبل مناقشة طرق اختيار النموذج، هناك شيء آخر نحتاج إلى مناقشته: تقييم النموذج. الغرض من **تقييم النموذج (model evaluation)** هو تقدير الخطأ العام للنموذج المحدد، أي إلى أي مدى يعمل النموذج المحدد على بيانات غير مرئية. من الواضح أن نموذج التعلم الآلي الجيد هو النموذج الذي لا يؤدي فقط أداءً جيدًا على البيانات التي تم تعلمها أثناء التدريب (وإلا، يمكن لنموذج التعلم الآلي بسهولة حفظ بيانات التدريب)، ولكن أيضًا على البيانات غير المرئية لها أداء جيد. لذلك، قبل نشر نموذج للإنتاج، يجب أن نتأكد نسبيًا من أن أداء النموذج لن ينخفض في مواجهة البيانات الجديدة.

لكن لماذا نحتاج إلى **التمييز بين اختيار النموذج وتقييم النموذج**؟ السبب هو أكثر من المناسب. إذا قمنا بتقدير خطأ التعميم للنموذج الذي اخترناه على نفس البيانات التي استخدمناها لتحديد النموذج المنتخب (على افتراض أن النموذج قد تم اختياره بناءً على مجموعة التدريب)، فسنحصل على تقدير متفائل. لماذا ؟ الجواب بسيط!! كان نموذج التعلم الآلي قادرًا على حفظ بيانات التدريب بسهولة. لذلك، لتقييم الأداء وتجنب مثل هذه المشاكل، نحتاج إلى بيانات مستقلة تمامًا لتقدير خطأ التعميم للنموذج.

تعتمد الإستراتيجية المقترحة لاختيار النموذج على كمية البيانات المتاحة.في حالة توفر الكثير من البيانات، قد نقوم بتقسيم البيانات إلى عدة أقسام، لكل منها غرض محدد. على سبيل المثال، لتعيين معاملات فائقة، قد نقسم البيانات إلى ثلاث مجموعات: **التدريب / التحقق من الصحة / الاختبار**. يتم استخدام مجموعة التدريب لتعليم نماذج مختلفة بعدد مختلف من مجموعات المعلمات الفائقة النموذجية. يتم تقييم هذه النماذج بعد ذلك على مجموعة التحقق ويتم اختيار

النموذج الذي يتمتع بأفضل أداءفي مجموعة التحقق من الصحة كنموذج منتخب. بعد ذلك، يتم إعادة تدريب النموذج على بيانات التدريب جنبًا إلى جنب مع بيانات التحقق من الصحة باستخدام مجموعة من المعاملات الفائقة المحددة، ويتم تقدير الأداء العام باستخدام مجموعة اختبار. إذاكان خطأ التعميم هذا مشابهًا لخطأ التحقق من الصحة، فإننا نعتقد أن النموذج سيعمل بشكل جيد على البيانات غير المرئية.

ما افترضناه ضمنيًا خلال المناقشة أعلاه هو أنه تم أخذ عينات من بيانات التدريب والتحقق من الصحة والاختبار من توزيع واحد. إذا لم يكن الأمر كذلك، فكل التقديرات خاطئة تمامًا. هذا هو السببفي أنه **من المهم التأكد من أن توزيع البيانات لا يتأثر بتجزئة بياناتك قبل إنشاء النموذج**.

ولكن ماذا لو كانت البيانات القليلة هي كل ما لدينا؟ كيف تختار وتقيم النموذجفي هذه الحالة؟ الإجابة هي أن تقييم النموذج لا يتغير لأننا ما زلنا بحاجة إلى مجموعة تجريبية يمكننا على أساسها تقدير خطأ التعميم للنموذج النهائي المحدد. ومن ثم، نقسم البيانات إلى مجموعتين، مجموعة للتدريب ومجموعة للاختبار. ما يتغير مقارنة بالطريقة السابقة هو كيف نستخدم البرنامج التدريبي. إحدى هذه التقنيات هي التحقق المتبادل، والتي ستتم مناقشتها لاحقًا. باختصار، ومع ذلك، فإن التحقق المتبادل هو أسلوب يقسم مجموعة التدريب الرئيسية إلى مجموعتين من التدريب والبيانات التجريبية (التحقق من الصحة).

عند التعامل مع البيانات المحددة زمنياً عندما يتعلق الأمر بالتنبؤ، يجب اختيار مجموعات التدريب والتحقق من الصحة والاختبار عن طريق تقسيم البيانات على طول محور الوقت. بمعنى، يتم استخدام البيانات "الأقدم" للتدريب، وأحدثها للتحقق من الصحة، وأحدثها للاختبار. أخذ العينات العشوائية لا معنى له في هذه الحالة.

تجزئة البيانات

على الرغم من أن خوارزميات التعلم الآلي تعتبر أدوات مذهلة وقوية في التنبؤ والتصنيف، فإن السؤال الذي يطرح نفسه هو مدى دقة هذه التنبؤات وهل هناك طريقة لقياس أداء النموذج؟ نظرًا لأن هذه الخوارزميات تحتوي على علامات مميزة للعينات، يمكن الإجابة على هذا السؤال بتقسيم عينات التدريب إلى عدة أقسام .

من خلال تقسيم البيانات، نقوم أولاً بإجراء التدريب على جزء من البيانات ، ثم نستخدم البيانات التجريبية لقياس كفاءة النموذج وإمكانية تعميمه. يشير **التعميم** إلى أداء النموذج في التعامل مع البيانات، وهو ما لم يلاحظه النموذج بعد في عملية التدريب. بالطبع في تصميم نماذج التعلم الآلي نقوم في أغلب الأحيان بتقسيم مجموعة بيانات

المشكلة إلى قسم آخر بالإضافة إلى بيانات تدريب والاختبار ، وطريقة هذا التقسيم على النحو التالي:

- **مجموعة تدريب:** عادة ما تكون أكبر مجموعات البيانات الثلاث وتستخدم للعثور على معاملات النموذج. تحدد مجموعة بيانات التدريب العلاقة الأساسية بين البيانات وعلاماتها بأفضل طريقة ممكنة.
- **مجموعة الاختبار:** قياس أداء النموذج بناءً على قدرة النموذج على التنبؤ بالبيانات التي لا تلعب دورًا في عملية التعلم، مجموعة الاختبار هي نفس البيانات التي لم يتم رؤيتها في عملية التعلم. هذه المجموعة تقيس أداء النموذج النهائي. إذا كان النموذج يعمل بشكل جيد في مجموعة التدريب ويناسب أيضًا مجموعة الاختبار، أي أنه يتنبأ بالتسمية الصحيحة لكمية كبيرة من بيانات الإدخال التي تم تجاهلها. وتجدر الإشارة إلى أن مجموعة الاختبار تستخدم عادة مرة واحدة فقط لتقييم أداء تعميم النموذج بشكل كامل بمجرد تحديد معاملات النموذج والمعاملات الفائقة بشكل كامل. ومع ذلك ، يتم استخدام مجموعة من عمليات التحقق لتقريب الأداء التنبئي لنموذج أثناء التدريب.
- **مجموعة التحقق من الصحة :** في تقييم أنواع مختلفة من النماذج والخوارزميات للمشكلة ، يتم استخدام مجموعة التحقق من الصحة. استخدم هذه البيانات لضبط المعاملات الفائقة ومنع النموذج من overfitting لتحديد أفضل نموذج.

لا توجد قواعد عامة حول كيفية مشاركة البيانات. ومع ذلك، يجب أن تكون مجموعة عمليات التحقق من الصحة كبيرة بما يكفى لتكون قادرًا على قياس فرق الأداء الذى نريد تحقيقه.

تُستخدم مجموعة التحقق من الصحة للحصول على قيم المعاملات الفائقة المثلى (تحسين المعاملات الفائقة) وللمساعدة فى تحديد النموذج، ويتم استخدام مجموعة الاختبار لتقييم أداء النموذج النهائى فى العينات التى تظهر فى عملية التعلم.

توازن التحيز والتباين (Bias-Variance Trade-Off)

في التعلم الآلي، تكون العلاقة بين تعقيد النموذج وخطأ التدريب والاختبار نتيجة لخاصيتين متنافستين هما التحيز والتباين. يشير التحيز إلى خطأ يحدث عند محاولة استخدام نموذج بسيط لحل مشكلة معقدةفي العالم الحقيقي. بمعنى آخر، هذا هو عدم قدرة نموذج التعلم الآلي على تصوير العلاقة الحقيقيةفي البيانات. على سبيل المثال، إذا أردنا استخدام الانحدار الخطي لتقدير

العلاقة غير الخطية، فسيكون للنموذج انحياز كبير. هذا لأن الخط المستقيم لا يمكن أن يكون مرنًا بما يكفي لتصوير علاقة غير خطية.

في المقابل، التباين هو الاختلاف في **الملاءمة** (**fit**) بين مجموعات البيانات. على سبيل المثال، عندما يكون النموذج مناسبًا، يكون له تباين كبير. لأن خطأ التنبؤ مختلف جدًا بالنسبة لمجموعة التدريب والاختبار.

بشكل عام، نود أن يكون لدينا أقل قدر ممكن من التحيز والتباين. ومع ذلك، فإن هذه المعايير لها تأثيرات معاكسة ولا يمكن تقليل التحيز دون زيادة التباين. من أجل إيجاد التوازن الأمثل بين التحيز والتباين، نقوم بتقييم عدة نماذج للعثور على أفضل المعاملات للنموذج. على سبيل المثال، نقوم أحيانًا بتقسيم مجموعة البيانات إلى جزأين: مجموعة التدريب ومجموعة الاختبار. عند تقييم أداء نموذج مبني على مجموعة التدريب، سواء في مجموعة التدريب أو في مجموعة الاختبار، نريد أن يكون خطأ التنبؤ منخفضًا قدر الإمكان. إذا كان النموذج يحتوي على خطأ تنبؤ منخفض في مجموعة التدريب، ولكن خطأ تنبؤ عالي في مجموعة الاختبار، يُقال إن النموذج به تباين كبير ونتيجة لذلك أدت البيانات إلى **الضبط الزائد (فرط التخصيص) (Overfitting)**. بشكل عام، النماذج الأكثر تعقيداً لها تباين أعلى. وذلك لأن النموذج المعقد يمكنه تتبع البيانات المحددة التي يتوافق معها بشكل أكثر دقة. ومع ذلك، نظرًا لأن النموذج المعقد يتتبع البيانات عن كثب، فمن المرجح أن يُظهر علاقة حقيقية في البيانات التدريبية وبالتالي أقل تحيزًا. لذلك، لا يمكن تحقيق اختيار نموذج ذي انحياز أقل نسبيًا إلا بتكلفة تباين أعلى.

من ناحية أخرى، إذا كان النموذج يحتوي على خطأ تنبؤ كبير في كل من المجموعات التدريبية والتجريبية، فيقال إن النموذج لديه تحيز كبير وبالتالي يتجاهل البيانات. باختصار، إذا كان نموذج بيانات التدريب معقدًا للغاية، فسيكون خطأ التنبؤ في بيانات التدريب منخفضًا. بمعنى آخر، يؤدي النموذج عادةً إلى التجهيز الزائد وبالتالي لا يتوافق جيدًا مع البيانات التجريبية، مما يؤدي إلى خطأ تنبؤ أعلى للمجموعة التجريبية. الهدف هو إيجاد الحل الأمثل، وهذا هو التوازن بين التحيز والتباين.

هناك عدة طرق لضبط التحيز والتباين. تحتوي معظم الخوارزميات على معاملات تنظم تعقيد النموذج. غالبًا ما يشار إلى هذه العملية باسم "**إعداد المعاملات الفائقة**"، والتي تعد جزءًا أساسيًا من مرحلة تقييم النموذج.

التحيز هو نموذج للتعلم الآلى للحصول على علاقة حقيقية بين متغيرات البيانات، هذا بسبب الافتراضات الخاطئة داخل خوارزمية التعلم. لا يولى نموذج التحيز المرتفع اهتمامًا كبيرًا لبيانات التدريب ويجعل النموذج بسيطًا للغاية ويؤدى دائمًا إلى خطأ كبير فى بيانات التدريب والاختبار.

يوضح التباين مدى تغير تقدير الوظيفة الموضوعية إذا تم استخدام بيانات تدريب مختلفة. بمعنى آخر، يعبر التباين عن مدى اختلاف متغير عشوائي عن قيمته المتوقعة. يولى نموذج التباين العالى الكثير من الاهتمام لبيانات التدريب ولا يعمم على البيانات التى لم يراها من قبل. نتيجة لذلك، تعمل هذه النماذج بشكل جيد جدًا على البيانات التدريبية، ولكن لديها معدل خطأ مرتفع فى البيانات التجريبية.

طرق التقييم

في حين أن تدريب النموذج هو خطوة أساسية، فإن كيفية تعميم النموذج على البيانات غير المرئية هي جانب مهم بنفس القدر يجب مراعاته بعد تصميم أي نموذج للتعلم الآلي. يجب التأكد من أن النموذج فعال حقًا وأن نتائج تنبؤاته يمكن الوثوق بها.

يمكن تدريب خوارزمية التصنيف على مجموعة بيانات محددة مع مجموعة فريدة من المعاملات التي يمكن أن تخلق حدود قرار تتناسب مع البيانات. نتيجة تلك الخوارزمية المعينة لا تعتمد فقط على المعاملات المتوفرة لتدريب النموذج، ولكن أيضًا على نوع بيانات التدريب. إذا كانت بيانات التدريب تحتوي على تباين بسيط أو كانت البيانات موحدة، فقد يؤدي النموذج إلى overfitting وإنتاج نتائج متحيزة على البيانات غير المرئية. لذلك، يتم استخدام طرق مثل التحقق المتبادل لتقليل overfittingفي التحقق من الصحة. التحقق من الصحة المتبادل هو أسلوب يقسم مجموعة التدريب الرئيسية إلى مجموعتين من التدريب والبيانات التجريبية (التحقق من الصحة). الطريقة الأكثر شيوعًا للتحقق المتبادل هي التحقق متعدد الأجزاء K-fold Validation، والذي يقسم مجموعة البيانات الرئيسية إلى أجزاء k متساوية الحجم. k هو رقم يحدده المستخدم ، وعادة ما يتم تحديد 5 أو 10.في هذه الطريقة،في كل مرة يتم استخدام واحدة من المجموعات الفرعية k كمجموعة تحقق (اختبار) ويتم تجميع المجموعة الفرعية k-1 معًا لتشكيل مجموعة تدريبية. للحصول على الكفاءة الكلية للنموذج، يتم حساب متوسط تقدير الخطأفي جميع التجارب.

فى تقنية التحقق من الصحة متعددة الأجزاءK-fold Validation ، يتم وضع كل جزء من البيانات مرة واحدة بالضبط فى مجموعة تجريبية ومرة واحدة فى مجموعة التدريب. هذا يقلل بشكل كبير من التحيز والتباين، لأنه يضمن أن كل مثيل من مجموعة البيانات الأصلية لديه فرصة للظهور فى مجموعة التدريب والتجريبية. إذا كانت لدينا بيانات إدخال محدودة، فإن التحقق من الصحة متعدد الأجزاء هو أحد أفضل الطرق لتقييم أداء النموذج.

معايير تقييم الأداء

لحساب معايير التقييم لنموذج التصنيف، نحتاج إلى أربع مجموعات من الفئة الحقيقية وفئة التنبؤ مع العناوين ، الموجبة الحقيقية ، الموجبة الخاطئة ، السلبية الحقيقية والسلبية الخاطئة، والتي يمكن تمثيلها في **مصفوفة الارتباك (Confusion Matrix)** (جدول 2ـ1):

- **موجب حقيقي (TP):** على سبيل المثال ، عندما كانت القيمة الفعلية للفئة "نعم" ، توقع النموذج أيضًا "نعم" (أي توقع صحيح).
- **موجب خاطئ (FP):** على سبيل المثال ، عندما كانت القيمة الفعلية للفئة "لا" لكن النموذج توقع "نعم" (أي توقع خاطئ).
- **منفى خاطئ (FN):** على سبيل المثال ، عندما تكون القيمة الفعلية للفئة "Yes" ، لكن النموذج توقع "لا" (أي توقع خاطئ).
- **منفى حقيقي (TN):** على سبيل المثال ، عندما تكون القيمة الفعلية للفئة "لا" وتوقع النموذج "لا" (أي ، كان التوقع صحيحًا).

جدول 2ـ1. مصفوفة الارتباك

		الفئة المتوقعة	
		مثبت	منفى
الفئة الحقيقي	مثبت	موجب حقيقي (TP)	منفى خاطئ (FN)
	منفى	موجب خاطئ (FP)	منفى حقيقي (TN)

المعيار الأكثر شيوعًا الذي يتم الحصول عليه من مصفوفة الارتباك هو دقتها **(accuracy)** أو عكسها: خطأ التنبؤ **(prediction error)**:

$$\text{الدقة} = \frac{TP + TN}{TP + FN + FP + TN}$$

$$\text{خطأالتنبؤ} = 1 - \text{الدقة}$$

الدقة هي نسبة عدد التنبؤات الصحيحة إلى العدد الإجمالي لعينات الإدخال.

عندما يتعين علينا تقييم نموذج ما، فإننا غالبًا ما نستخدم معدلات الخطأ والدقة، ولكن ما نركز عليه بشكل أساسي هو مدى موثوقية نموذجنا، وكيف يعمل على مجموعة بيانات مختلفة (قابلية التعميم) ومدى مرونته. لا شك أن الدقة معيار مهم للغاية يجب أخذهفي الاعتبار، لكنها لا تقدم دائمًا صورة كاملة لأداء النموذج.

عندما نقول إن النموذج موثوق، فإننا نعني أن النموذج قد حصل على البيانات بشكل صحيح ووفقًا لطلب التعلم. لذلك، فإن التنبؤات التي قدمتها قريبة من القيم الفعلية.في بعض الحالات، قد يؤدي النموذج إلى دقة أفضل، لكنه قد لا يفهم البيانات بشكل صحيح وبالتالي يؤدي بشكل سيئ عندما تكون البيانات مختلفة. هذا يعني أن النموذج ليس موثوقًا وقويًا بدرجة كافية وبالتالي يحد من استخدامه.

على سبيل المثال، لدينا 980 تفاحة و20 برتقالة ولدينا نموذج يصنف كل فاكهة على أنها تفاحة. لذلك، دقة النموذج 98٪ = 180/9، وبناءً على معيار الدقة لدينا نموذج دقيق للغاية. ومع ذلك، إذا استخدمنا هذا النموذج للتنبؤ بالثمار المستقبلية، فسوف نفشل. لأن هذا النموذج يمكن أن يتنبأ بفئة واحدة فقط.

الحصول على صورة كاملة للنموذج، على سبيل المثال كيف يدرك البيانات وكيف يمكن التنبؤ بها، يساهمفي فهمنا العميق للنموذج ويساعد على تحسينه. لذا، افترض أن لديك نموذجًا يحقق دقة 90٪، فكيف يمكنك تحسينه؟ لتصحيح الخطأ، يجب أن ندركه أولاً. وبالمثل، لتحسين النموذج، نحتاج إلى النظرفي كيفية عمل النموذج على مستوى أعمق. ومع ذلك، لا يتم تحقيق ذلك بمجرد النظر إلى معيار الدقة، وبالتالي يتم النظرفي معايير أخرى. معايير مثل **الدقة (precision) والاستدعاء (recall)** و **F1-Score** هي أمثلة على هذه المعايير.

يشير الاستدعاء إلى قدرة النموذج على التنبؤ بالحالات الإيجابية من بين جميع الإيجابيات الحقيقية. من ناحية أخرى، تقيس الدقة جزء الإيجابيات الحقيقية بين العينات التي يُتوقع أن تكون إيجابية. قد لا تكون الدقة والاستدعاء وحدهما مناسبين لتقييم النموذج، لذلك يتم استخدام درجة F1، والتي تتضمن الدقة والاستدعاء، وتشير إلى مدى دقة المصنف. كلما زادت درجة F1، كان أداء نموذجنا أفضل. طريقة حساب هذه المعايير على النحو التالي:

$$\text{الاستدعاء} = \frac{TP}{FN + TP}$$

$$\text{الدقة} = \frac{TP}{FP + TP}$$

$$\text{F1 درجة} = 2 \times \frac{\text{الاستدعاء} * \text{الدقة}}{\text{الاستدعاء} + \text{الدقة}}$$

أدوات ومكتبات بايثون

تثبيت بايثون

في هذا القسم، نقدم خطوات تثبيت Pythonفي نظام التشغيل Windows. نظرًا لعدم وجود بيئة Python مضمنةفي نظام التشغيل Windows، يجب تثبيتها بشكل مستقل. يمكن تنزيل حزمة التثبيت من موقع Python الرسمي (www.python.org). بعد فتح الموقع الرسمي، ابحث عن شريط التنقل الذي يحتوي على زر تنزيل (download). يوصي موقع الويب برابط افتراضيًا، حيث يمكنه تحديد نظام التشغيل الخاص بك والتوصية بأحدث إصدار من Python 3.x. بعد الدخول إلى صفحة التنزيل الخاصة بالإصدار ذي الصلة، توجد مقدمة أساسية حول البيئة التي تريد تنزيلها. تم تصميم العديد من الإصدارات المختلفة بشكل أساسي لأنظمة التشغيل المختلفة. اعتمادًا على ما إذا كان النظام 32 بت أو 64 بت، يمكنك تحديد ملفات مختلفة لتنزيلها.في الصفحة الجديدة التي تفتح، يمكننا العثور على إصدارات أخرى، بمافي ذلك أحدث إصدار تجريبي والإصدار المطلوب. إذا كنت تريد تثبيت الإصدار 64 بت 3.9.6، فانقر فوق الارتباط الموجودفي الصفحة الحالية.

بعد تحميل Python، حان وقت تثبيته. يعد تثبيت حزمة Windows أمرًا سهلاً للغاية. تمامًا مثل تثبيت برامج Windows الأخرى، نحتاج فقط إلى تحديد الخيار المناسب والنقر فوق الزر "التالي" لإكمال التثبيت. عندما تظهر الخيارات أثناء التثبيت، لا تتسرعفي الخطوة التالية. لأنه من أجل الراحةفي المستقبل، عليك اختيار زر "Add Python 3.9.6 to PATH" لمتغير البيئة، يمكن تنفيذ أوامر Python مباشرة وبسهولة على سطر أوامر Windowsفي المستقبل. بعد تحديد "Add Python 3.9.6 to PATH”، حدد التثبيت المطلوب. بالطبع، من الممكن أيضًا تحديد موقع التثبيت، المثبتفي الدليل C على محرك الأقراص C افتراضيًا. ومع ذلك، من الأفضل معرفة ما هو دليل المستخدم حتى تتمكن من العثور على ملفات Python.exe المثبتة عند الحاجة. اتبع التعليمات لتثبيت Python بنجاح على نظامك.

ابدأ مع بايثون

هناك طريقتان لتشغيل بايثون:

1) **استخدام IDLE الخاص ببايثون.** إذاكنت ترغب في تشغيل Python ، يمكنك النقر فوق الزر "ابدأ" على سطح مكتب Windows واكتب "IDLE" في مربع "بحث" للدخول بسرعة إلى "read-evaluate-print-loop". بعد تشغيل البرنامج ، سترى صورة مثل الصورة أدناه:

```
Python 3.6.7 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.7 (v3.6.7:6ec5cf24b7, Oct 20 2018, 13:35:33) [MSC v.1900 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
Ln: 3  Col: 4
```

IDLE هو بيئة تطوير متكاملة (Integrated Development Environment) توفر محرر واجهة مستخدم رسومي لـ Python. يبدو تشغيله بسيطًا ومناسبًا للمبتدئين في تعلم لغة Python. يوفر IDLE بيئة REPL، أي أنه يقرأ ويقيم ويحسب مدخلات المستخدم ()، ثم يطبع النتيجة ()، وتظهر رسالة "حلقة" (في انتظار الإدخال التالي).

2) **باستخدام Windows Prompt.** هناك طريقة أخرى لتشغيل Python وهي تشغيل برامج Python من سطر أوامر Windows. للقيام بذلك ، اضغط على مفتاحي "Win + R" لفتح مربع الإشعارات ، ثم أدخل "cmd" في المربع الذي يفتح. إذا قمت بتحديد "إضافة Python 3.x إلى PATH" عند تثبيت Python ، فقد تمت إضافة Python المثبت إلى متغير بيئة Windows. الآن بإدخال كلمة "python" بعد ظهور ">>>" Python ، سيتم تشغيلها بنجاح وسترى صورة مثل الصورة أدناه:.

```
Administrator: Command Prompt - python
Microsoft Windows [Version 10.0.18363.628]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>python
Python 3.6.7 (v3.6.7:6ec5cf24b7, Oct 20 2018, 13:35:33) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

يشير الإشعار ">>>" إلى أن التثبيت باستخدام Python كان ناجحًا وأن Python قد بدأت في العمل.

تثبيت المكتبات

يجب عليك استخدام pip لإدارة مكتبات pip .pip هي أداة أساسية تتيح لك تنزيل الحزم التي تحتاجها وتحديثها وحذفها. بالإضافة إلى ذلك، يمكن استخدامه للتحقق من التبعيات المناسبة والتوافق بين الإصدارات.

يتم تثبيت مكتبة باستخدام pip في سطر أوامر Windows. على سبيل المثال، افترض أننا نريد تثبيت مكتبة NumPy. توضح الخطوات التالية كيفية تثبيت هذه المكتبة:

- اضغط أولاً على مفتاحي "Win + R" لفتح مربع الإشعارات ثم أدخل "cmd" في المربع الذي يفتح. ثم أدخل الأمر التالي في سطر الأوامر:

```
> pip install numpy
```

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.18363.628]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>pip install numpy
```

- للتأكد من تثبيت المكتبة ، قم بتشغيل سطر أوامر Python واكتب الأمر التالي::

```
>>> import numpy
```

- لن يتم عرض أي رسالة إذا تم تثبيت المكتبة بشكل صحيح. إذا لم يتم تثبيت المكتبة على جهاز الكمبيوتر الخاص بك ، فسترى هذه الرسالة عن طريق تنفيذ الأمر أعلاه:

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named numpy
```

Jupyter Notebook

يعد Jupyter Notebook أداة قوية بشكل لا يصدق لتطوير مشاريع التعلم الآلي وتقديمها بشكل تفاعلي، والتي يمكن أن تتضمن نصًا أو صورة أو صوتًا أو فيديو بالإضافة إلى تنفيذ التعليمات البرمجية. يجمع Notebook بين التعليمات البرمجية والمخرجات مع الرسوم التوضيحية والنص السردي والمعادلات الرياضية والوسائط الأخرى في مستند واحد. بمعنى آخر، يعد Notebook مستندًا واحدًا يمكنك من خلاله تنفيذ التعليمات البرمجية وعرض المخرجات وإضافة الأوصاف والصيغ والرسوم التخطيطية لجعل عملك أكثر وضوحًا وقابلية للفهم وقابلية للتكرار والمشاركة.

لتثبيت Jupyter Notebook، يجب أن يكون لديك Python مثبتًا مسبقًا. حتى إذا كنت تخطط لاستخدام Jupyterللغات البرمجة الأخرى، فإن بايثون هي الركيزة الأساسية لـ Jupyter. لتثبيت Jupyter، ما عليك سوى كتابة الأمر التالي في سطر أوامر Windows:

```
> pip install jupyter
```

لتشغيل Jupyter، افتح سطر الأوامر واكتب الأمر التالي فيه:

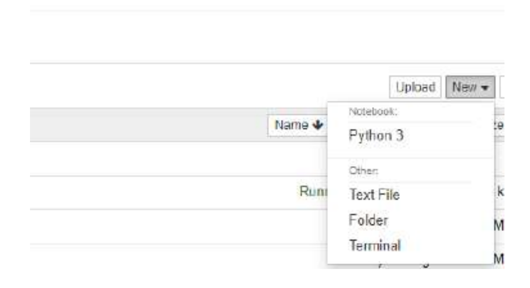
```
> jupyter notebook
```

بعد تنفيذ الأمر أعلاه، سيتم تشغيل متصفح الويب الافتراضي الخاص بك باستخدام Jupyter. عند تشغيل Jupyter Notebook، انتبه إلى دليل سطر الأوامر، حيث يصبح هذا الدليل هو الدليل الرئيسي الذي يظهر على الفور على Jupyter Notebook وسيكون لديك حق الوصول إلى الملفات والأدلة الفرعية الموجودة فيه فقط. قم بتشغيل أمر دفتر الملاحظات Jupyter Notebook وسترى صفحة مثل الصفحة أدناه:

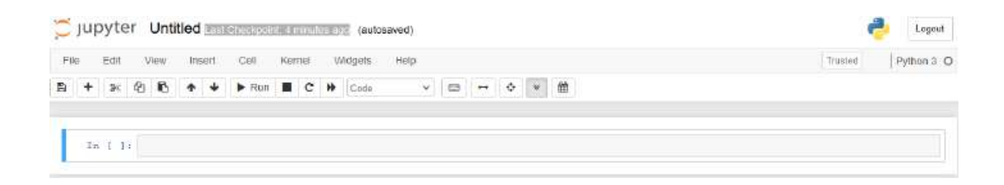


ومع ذلك، فإن هذه الصفحة ليست Notebook بعد وهي سطح المكتب الوحيد من Jupyter المصمم لإدارة دفاتر ملاحظات Jupyter الخاصة بك واستخدامها كدليل لاستكشاف وتحرير وإنشاء Notebook الخاصة بك. تعتمد أجهزة الكمبيوتر المحمولة وأجهزة سطح المكتب منJupyter على المستعرض، ويقومJupyter بإعداد خادم Python محلي لتوصيل هذه التطبيقات بمتصفح الويب الخاص بك.

لإنشاء Notebook جديد، انتقل إلى الدليل حيث تريد إنشاء اول Notebook لك وانقر على زر القائمة المنسدلة "جديد" أعلى يمين سطح المكتب وحدد "Python 3".:



بعد ذلك، سيتم فتح دفتر ملاحظاتك الأولفي علامة تبويب جديدة كما هو موضح أدناه:



إذا عدت إلى سطح مكتب Jupyter ، فسترى ملف Untitled.ipynb الجديد وسترى نصًا أخضر يخبرك أن دفتر ملاحظاتك قيد التشغيل.

نختبر كيفية تنفيذ خلية بمثال كلاسيكي: اكتب print ('Hello World!') في خلية وانقر على الزر ▶ Run في شريط الأدوات أعلاه، أو اضغط على Ctrl + Enter. ستكون النتيجة على هذا النحو:

```
In [1]: print('Hello World!')

        Hello World!
```

Colab

Colaboratory ، أو Colab ، عبارة عن منتج بحثي من Google (خدمة سحابية) يسمح للمطورين بكتابة وتنفيذ كود Python من خلال متصفحهم. يعد Google Colab أداة رائعة لمهام التعلم العميق ويساعد على تطوير النماذج باستخدام مكتبات متعددة مثل Keras و Pytorch و OpenCv و Tensorflow والمزيد. Colab عبارة عن دفتر ملاحظات يستند إلى Jupyter ولا يحتاج إلى تثبيت ولديه إصدار مجاني رائع يوفر وصولاً مجانيًا إلى موارد الحوسبة من Google مثل GPU و TPU.



لماذا Colab؟

يعتبر Colab مثاليًا لكل شيء بدءًا من تحسين مهارات تشفير Python إلى العمل مع مكتبات التعلم العميق ، مثل PyTorch و Keras و TensorFlow و OpenCV. يمكنك إنشاء وتحميل وحفظ ومشاركة دفاتر الملاحظات في Colab، وتثبيت Google Drive الخاص بك واستخدام كل ما قمت بتخزينه هناك ، وتحميل دفاتر الملاحظات مباشرة من GitHub ، وتحميل ملفات Kaggle، ودفاتر الملاحظات الخاصة بك، والمشاركة والقيام بأي شيء آخر قد ترغب في القيام به.

ميزة أخرى رائعة في Google Colab هي ميزة التعاون. إذا كنت تعمل مع مطورين متعددين في مشروع ما، فإن استخدام نوتبوك Google Colab يعد أمرًا رائعًا. تمامًا مثل التعاون في

مستند Google Docs ، يمكنك البرمجة مع مبرمج متعدد باستخدام دفتر ملاحظات Colab.
بالإضافة إلى ذلك ، يمكنك أيضًا مشاركة عملك المكتمل مع مطورين آخرين.
باختصار ، يمكن سرد الأسباب المختلفة لاستخدام Colab على النحو التالي:

- مكتبات مثبتة مسبقًا.
- مخزنة في السحابة.
- التعاون.
- استخدام GPU و TPU مجاني.

ومع ذلك، هناك سيناريوهان يجب عليك استخدام Jupyter Notebookفي جهازك:

1. تجنب Google Colab إذا كنت تهتم بالخصوصية وترغب في الحفاظ على سرية اكوادك.
2. إذا كان لديك جهاز قوي بشكل لا يصدق مع امكانية GPU و TPU.

اعداد Google Colab

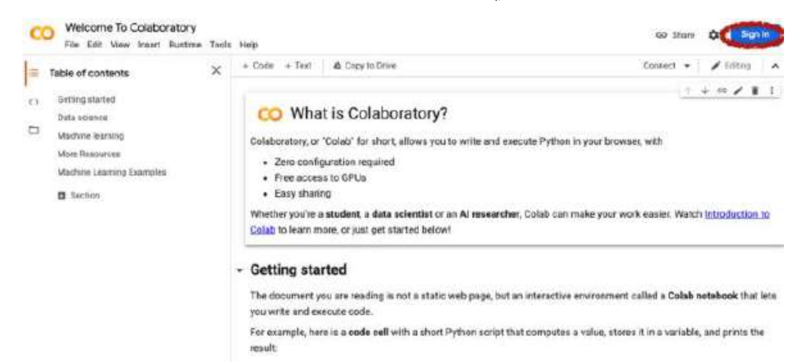
تعد عملية إعداد Colab سهلة نسبيًا ويمكن إكمالها بالخطوات التالية على أي نوع من الأجهزة:

1. **قم بزيارة صفحة Google Colab:**

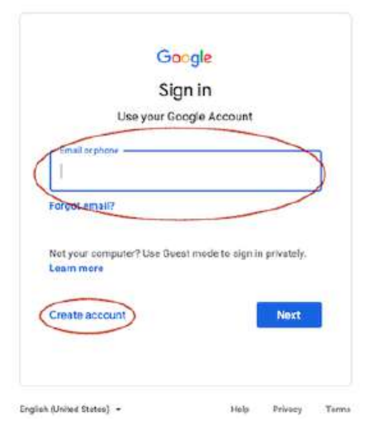
http://colab.research.google.com

سينقلك تحميل الموقع أعلاه إلى صفحة الترحيب الخاصة بـ Google Colaboratory

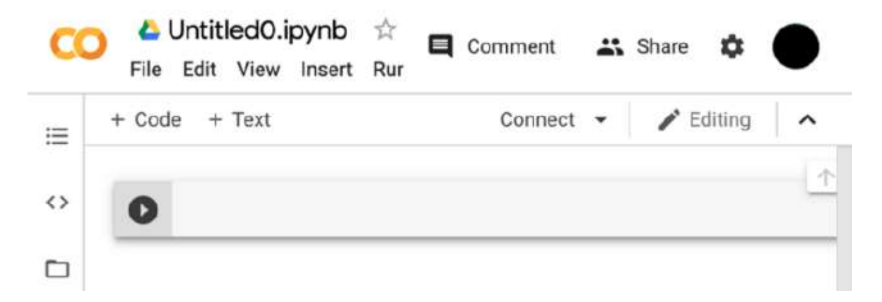
2. **انقر فوق الزر تسجيل الدخول في الجزء العلوي الأيمن:**



3. **قم بتسجيل الدخول باستخدام حساب Gmail الخاص بك. إذا لم يكن لديك حساب Gmail ، فقم بإنشاء حساب:**



4. **بمجرد الانتهاء من تسجيل الدخول ، تكون جاهزًا لاستخدام Google Colab.**
5. **من خلال النقر فوق ملف ==> دفتر ملاحظات جديد، يمكنك بسهولة إنشاء دفتر ملاحظات جديد من Colab في هذه الصفحة.**



أطر التعلم العميق

يعد تطوير شبكة عصبية عميقة وإعدادها لحل المشكلات مهمة صعبة للغاية. لأن العديد من القطع تحتاج إلى تجميعها معًا لإنشاء تدفق منظم وتنظيمه من أجل تحقيق الأهداف التي نريد تحقيقها من خلال التعلم العميق. لذلك، يحتاج الباحثون أو علماء البيانات إلى إطار عمل لتمكين حلول أسهل وأسرع وأفضل جودة للتجارب والأبحاث. تساعد هذه الأطر الباحثين والمطورين على التركيز على المهام الأكثر أهمية، بدلاً من استثمار المزيد من الوقتفي العمليات الأساسية. توفر أطر ومنصات التعلم العميق فكرة مجردة إلى حد ما عن المهام المعقدة مع وظائف بسيطة يمكن استخدامها من قبل الباحثين والمطورين كأداة لحل المشكلات الأكبر.

باي تورج (PyTorch)

PyTorch هي بيئة تعلم آلي تعتمد على Torch وهي مثالية لتصميم الشبكة العصبية. تم تطوير PyTorch بواسطة مختبر أبحاث الذكاء الاصطناعي في Facebook، وتم إصداره في يناير

2016 كمكتبة مجانية ومفتوحة المصدر، ويستخدم بشكل أساسي في رؤية الكمبيوتر والتعلم العميق وتطبيقات معالجة اللغة الطبيعية، ويدعم تطوير البرامج المستندة إلى السحابة. يعد تنفيذ شبكة عصبية في PyTorch أسهل وأكثر سهولة من البيئات الأخرى. مع دعم وحدة المعالجة المركزية ووحدة معالجة الرسومات، يمكن تدريب الشبكات العصبية العميقة المعقدة بمجموعات بيانات كبيرة.

المزايا والعيوب

المزايا

- سهولة التعلم.
- مرن وسريع.
- سهولة تصحيح الاخطاء.

العيوب

- عدم وجود أدوات دعم الرسم والتوضيح مثل tensorboard

تنسرفلو (TensorFlow)



TensorFlow هي واحدة من أشهر بيئات التعلم الآلي والتعلم العميق التي يستخدمها المطورون والباحثون. تم إطلاق TensorFlow لأول مرة بواسطة فريق Google Brain في عام 2007 ويمكن تشغيله على وحدات المعالجة المركزية ومسرعات الذكاء الاصطناعي المتخصصة، بما في ذلك وحدات معالجة الرسومات (GPU) و (TPU). يتوفر TensorFlow على أنظمة Linux و macOS و Windows 64 بت ومنصات الحوسبة المحمولة، بما في ذلك Android و iOS. يمكن نشر النماذج المدربة على TensorFlow على أجهزة سطح المكتب والمتصفحات وحتى وحدات التحكم الدقيقة. هذا الدعم الشامل يجعل TensorFlow فريدًا وجاهزًا للانطلاق. سواء كنت تعمل على رؤية الكمبيوتر أو معالجة اللغة الطبيعية أو نماذج السلاسل الزمنية ، فإن TensorFlow عبارة عن منصة تعلم آلي قوية وعالية الأداء.

المزايا والعيوب

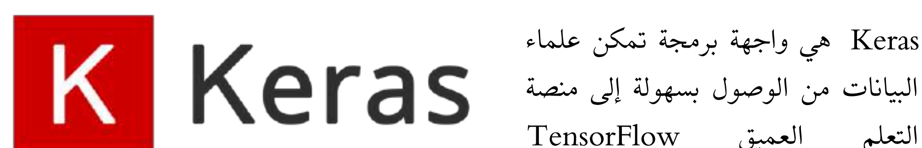
المزايا

- دعم ممتاز للرسوم البيانية الحسابية، لكل من الحوسبة والتوضيح.
- يمكن تثبيت TensorFlow على أجهزة سطح المكتب والمتصفحات وحتى وحدات التحكم الدقيقة.

العيوب

- منحنى التعلم منحدر بسبب واجهات برمجة التطبيقات API منخفضة المستوى (صعوبة التعلم).
- قد يكون فهم بعض رسائل الخطأ في TensorFlow أمراً صعباً للغاية.

كيراس (Keras)



Keras هي واجهة برمجة تمكن علماء البيانات من الوصول بسهولة إلى منصة التعلم العميق TensorFlow واستخدامها. إنها واجهة برمجة تطبيقات (API) ومساحة عمل تعلم عميق مفتوحة المصدر مكتوبة بلغة Python تعمل على TensorFlow وتم دمجها الآن في النظام الأساسي. دعم Keras سابقًا العديد من (backend)، ولكن مع إطلاق الإصدار 2.4.0 في يونيو 2020 ، فهو مرتبط حصريًا بـ TensorFlow. تم تصميم Keras ، باعتبارها API عالية المستوى، لإجراء تجارب سريعة وسهلة تتطلب ترميزًا أقل من خيارات التعلم العميق الأخرى. الهدف هو تسريع تنفيذ نماذج التعلم الآلي، ولا سيما الشبكات العصبية العميقة، من خلال عملية تطوير "سرعة التكرار العالية". يمكن تشغيل طرازات Keras على وحدة المعالجة المركزية أو وحدة معالجة الرسومات ونشرها على منصات متعددة، بما في ذلك متصفحات الويب وأجهزة Android و iOS المحمولة. تعد Keras أبطأ من TensorFlow و PyTorch ولكنها تتميز ببنية بسيطة وهي أكثر قابلية للقراءة وموجزة وسهلة الاستخدام وقابلة للتطوير. يعد Keras أكثر ملاءمة لمجموعات البيانات الصغيرة ويوصى به للمبتدئين نظرًا لتصميمه البسيط والمفهوم.

المزايا والعيوب

المزايا

- API عالي المستوى.
- سهلة التعلم.
- سهولة إنتاج النماذج.
- سهل الاستخدام.
- لايحتاج إلى خلفية قوية في التعلم العميق.

العيوب

- مناسب لمجموعة البيانات الصغيرة.
- في بعض الاحيان, بطيء مع GPU.

إذا كنت مبتدئًا أو تجرب نماذج بسيطة للمشكلة، فإن Keras هو الخيار الأفضل لك. لأنه من الأسهل البدء بها، فمن السهل جداً ألا تحتاج إلى معرفة أي شيء عن التعلم العميق لتعليم شبكة عصبية عميقة!! على الرغم من أن PyTorch حققت نتائج جيدة في هذا الصدد (مقارنةً بـ Tensorflow) ، إلا أن Keras أفضل.

خلاصة الفصل

- تشير البيانات إلى أجزاء مميزة من المعلومات التي عادة ما يتم تنسيقها وتخزينها لتناسب غرضًا محددًا.
- يتم تمثيل كل نقطة بيانات غالبًا بواسطة متجه سمة، ويمثل كل إدخال في المتجه سمة.
- يمكن تصنيف البيانات على أنها مقروءة آليًا أو قابلة للقراءة البشرية أو كليهما.
- يعد التعلم الخاضع للإشراف أحد أكثر فروع التعلم الآلي استخدامًا ، والذي يستخدم بيانات التدريب المصنفة لمساعدة النماذج على إجراء تنبؤات دقيقة.
- في التصنيف، يتم تعريف الفئات مسبقًا وغالبًا ما يشار إليها على أنها أهداف أو علامات أو فئات.
- الانحدار هو عملية إحصائية تجد علاقة ذات دلالة إحصائية بين المتغيرات التابعة والمستقلة، وكخوارزمية، تتنبأ برقم مستمر.
- يحدث التعلم الآلي غير الخاضع للإشراف في حالة عدم وجود تصنيف للبيانات.

اختبار

1. قم بتسمية الطرق المختلفة للتعلم الآلي، ووصف الاختلافات والمزايا والعيوب لكل منها؟
2. ما المقصود بالتعميم في التعلم الآلي؟
3. ما هو الفرق بين المعاملات والمعاملات الفائقة؟
4. لماذا يتم تقسيم البيانات لتعليم خوارزمية التعلم الآلي؟
5. هل يمكن أن يكون النموذج الذي حقق دقة 98٪ في مجموعة التدريب ولكن بدقة 79٪ في مجموعة الاختبار نموذجًا مقبولاً أم لا؟ إعطي سبباً.
6. لنفترض أن نموذجًا في مجموعة بيانات ذات فئتين مختلفتين وغير متوازنتين للغاية يحقق دقة تصل إلى 99٪ ، فهل يمكننا القول أن هذا النموذج يتمتع بكفاءة عالية جدًا بناءً على معيار الدقة فقط؟ اشرح السبب.
7. لماذا يتم استخدام مجموعة بيانات التحقق من الصحة؟
8. ما الذي يسبب الضبط الزائد Overfitting؟
9. إلى ماذا يشير التباين العالي والتحيز العالي؟

3

الشبكات العصبية امامية التغذية

اهداف التعليم:

- التعرف على شبكة بيرسيبترون.
- التعرف على الشبكة العصبية امامية التغذية.
- التحسين.
- دالة الخطأ.
- تنفيذ شبكة عصبية في keras.

المقدمة

في هذا الفصل، نقدم بنية الشبكات العصبية، ونصف بالتفصيل كيف تعمل الخلايا العصبية، ثم نصف عملية التدريبفي الشبكات العصبية والمفاهيم الموجودةفي هذا المجال. هذه المفاهيم بمثابة أساس للفصول القادمة.

الشبكات العصبية الاصطناعية (Artificial Neural Networks)

الشبكات العصبية الاصطناعية، أو باختصار **ANN**، هي مجموعة من نماذج التعلم الآلي المستوحاة بشكل عام من دراسات الجهاز العصبي المركزي للثدييات. بمعنى آخر، إنها نموذج حسابي يحاكي كيفية عمل الخلايا العصبيةفي دماغ الإنسان. تتكون كل شبكة عصبية اصطناعية من عدة "**عصبونات**" متصلة منظمةفي "**طبقات**". ترسل الخلايا العصبيةفي كل طبقة رسائل إلى الخلايا العصبيةفي الطبقة التالية.

الشبكة العصبية الاصطناعية هي محاولة لمحاكاة شبكة من الخلايا العصبية التي تشكل دماغ الإنسان حتى تتمكن أجهزة الكمبيوتر من التعلم واتخاذ القرارات بطرق بشرية.

تحتوي الشبكة العصبية الاصطناعية على طبقة إدخال وطبقة إخراج وطبقة مخفية واحدة أو أكثر مترابطة. تتكون الطبقة الأولى من الخلايا العصبية المدخلة. ترسل هذه الخلايا العصبية البيانات إلى طبقات أعمق. تتلقى كل طبقة بعد طبقة الإدخال، بدلاً من الإدخال الأولي، إخراج الطبقة السابقة كمدخل. أخيرًا، ينتج آخر طبقة إخراج للنموذج.

تحدد أمثلة التدريب بشكل مباشر المخرجات التي يجب إنشاؤها لكل إدخال x. تحاول طبقة المخرجات حساب قيمة قريبة من ناتج معين لعينات تدريب مماثلة. ومع ذلك، فإن سلوك الطبقات الداخلية لا يتأثر بشكل مباشر بعينات التدريب ، وخوارزمية التدريب هي التي تحدد كيفية عمل هذه الطبقات من خلال اتخاذ قراراتهم الخاصة لإنتاج المخرجات المرغوبة. ونتيجة لذلك، فإن وظيفة الطبقات الداخلية، بناءً على المخرجات المرغوبة التي تم الحصول عليها من عينات التدريب، غير محددة بوضوح وتعمل كصندوق أسود، ومن ثم تسمى هذه الطبقات بالطبقات المخفية.

تتمثل مهمة الطبقات المخفية في تحويل الإدخال إلى شيء يمكن لطبقة المخرجات استخدامه. مع زيادة عدد الطبقات المخفية، ننتقل إلى شبكة أعمق لديها القدرة على حل مشاكل أكثر تعقيداً من نظيراتها الضحلة.

لقد تعلم علماء الأعصاب الإدراكيون الكثير عن الدماغ البشري منذ أن طور علماء الكمبيوتر لأول مرة الشبكة العصبية الاصطناعية الأصلية. كان أحد الأشياء التي تعلموها أن أجزاء مختلفة

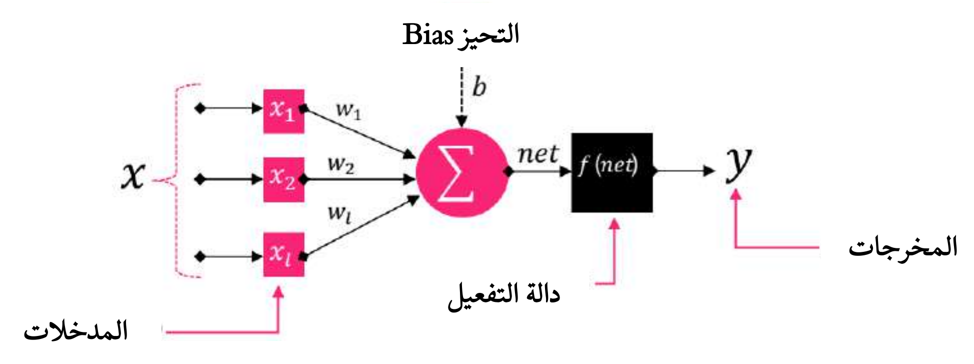
من الدماغ مسؤولة عن معالجة جوانب مختلفة من المعلومات، وهذه الأجزاء مرتبة بشكل هرمي. وهكذا، تدخل المدخلات إلى الدماغ، وكل مستوى من الخلايا العصبية يوفر البصيرة، ثم يتم نقل المعلومات إلى المستوى الأعلى التالي. هذه هي الآلية التي تحاول ANN تكرارها.

تعد الشبكات العصبية الاصطناعية رائعة لتعدد استخداماتها، مما يعني أنها تتحسن من خلال التعلم من البيانات واكتساب المزيد من المعلومات في العروض اللاحقة.

لكي تتمكن الشبكات العصبية الاصطناعية من التعلم، يجب أن يكون لديها قدر هائل من البيانات، وهو ما يسمى مجموعة التدريب. عندما تريد تعليم الشبكات العصبية الاصطناعية كيفية التمييز بين القط والكلب، يوفر البرنامج التعليمي آلاف الصور التي تم وضع علامات عليها للكلاب لكي تبدأ الشبكةفي التعلم. عند تدريبه على قدر كبير من البيانات، يحاول تصنيف البيانات المستقبلية إلى فئات مختلفة بناءً على ما يعتقد أنه يراه (أو يسمعه، اعتمادًا على مجموعة البيانات). خلال فترة التدريب، تتم مقارنة إخراج الجهاز مع الأوصاف البشرية (العلامات) لما يجب مشاهدته. إذا كانت متطابقة، فإن النموذج يعمل بشكل جيد. إذا كان غير صحيح، فإنه يستخدم **الانتشار الخلفي backpropagation** لضبط التعلم.

بيرسيبترون (perceptron)

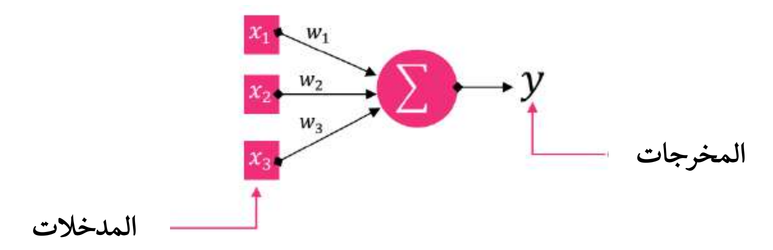
الخلايا العصبية هي عنصر أساسي في أي شبكة عصبية اصطناعية. يُطلق على أبسط نوع من نمذجة الخلايا العصبية اسم **بيرسيبترون Perceptron** ، والذي يمكن أن يحتوي على عدد كبير من المدخلات بمخرج واحد. يوضح الشكل 3ـ1 مخططًا لبيرسيبترون. يستخدم Perceptron التعلم الخاضع للاشراف لتصنيف أو توقع المخرجات. يصنف المستشعر أحادي الطبقة البيانات عن طريق رسم **حدود القرار (decision boundary)** باستخدام خط محدد.



شكل 3ـ1. بيرسيبترون

دعونا نلقي نظرة على كيفية عمل بيرسبترون. يعمل Perceptron عن طريق التقاط بعض المدخلات العددية جنبًا إلى جنب مع ما يعرف باسم **الأوزان (weights) والتحيزات او**

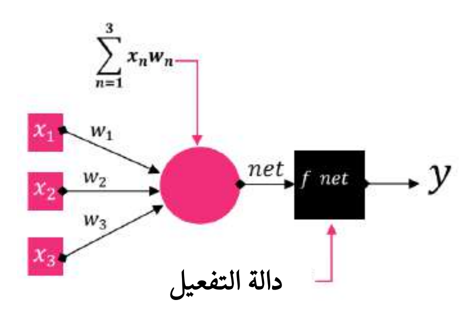
الانحيازات (bias). ثم يقوم بضرب هذه المدخلات بالأوزان المقابلة (المعروفة باسم مجموع الأوزان). ثم يتم إضافة هذا الناتج مع التحيز. تأخذ **دالة التنشيط(Activation Function)** مجموع الاوزان والانحياز كمدخلات وترجع الناتج النهائي. كان الأمر محيرًا !!!! ... دعونا نحلل Perceptron ، لتحسين طريقة عمله. يتكون بيرسيبترون (الشكل 3ـ1) من أربعة أجزاء رئيسية: **قيم الإدخال والأوزان والتحيز (الانحياز) وإجمالي الوزن ودالة التنشيط**. افترض أن لدينا خلية عصبية وثلاثة مدخلات x_1، x_2، x_3 مضروبة في الأوزان w_1، w_2، w_3 على التوالي:



الفكرة بسيطة ، بالنظر إلى القيمة العددية للمدخلات والأوزان، فهناك دالة داخل الخلية العصبية تنتج مخرجات. **الآن السؤال ما هي هذه الدالة؟** هذه الدالة تعمل كالاتي:

$$y = x_1 w_1 + x_2 w_2 + x_3 w_3$$

هذه الدالة تسمى **مجموع الاوزان** ، لأنها مجموع الأوزان والمدخلات. هذا جيد حتى الآن ، لكن إذا أردنا أن تكون المخرجات في نطاق معين، على سبيل المثال من 0 إلى 1 ، فماذا يجب أن نفعل؟! يمكننا القيام بذلك باستخدام ما يسمى **بدالة التنشيط (دالة التفعيل)**. دالة التنشيط هي دالة تحول إدخالًا معينًا (في هذه الحالة ، مجموع الاوزان) إلى إخراج محدد ، بناءً على مجموعة من القواعد:



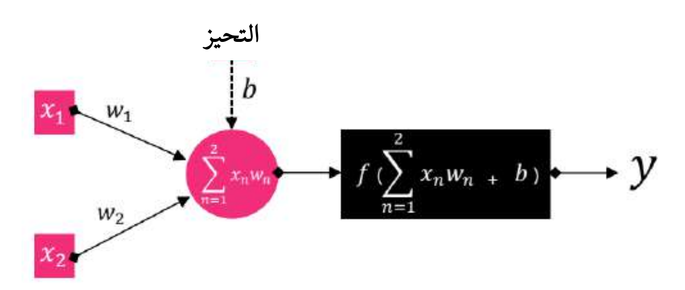
لدينا الآن كل ما نحتاجه تقريبًا لبناء Perceptron. آخر شيء هو التحيز. التحيز هو معامل إضافي في الشبكة العصبية يُستخدم لتنظيم الإخراج جنبًا إلى جنب مع الوزن الإجمالي للمدخلات العصبية. بالإضافة إلى ذلك، يسمح مقدار التحيز لدالة التنشيط بالتغيير إلى اليمين أو اليسار. أسهل طريقة لفهم التحيز من خلال الثابت c هي دالة خطية:

$$y = mx + c$$

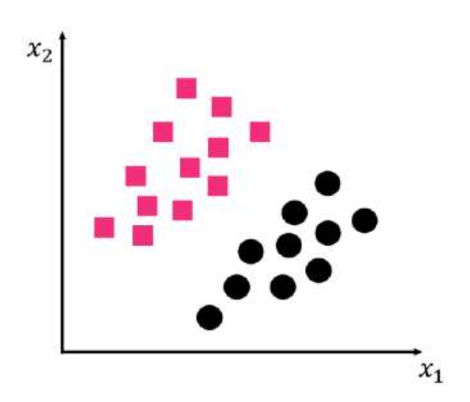
يتيح لك ذلك التمرير لأعلى ولأسفل في السطر لمطابقة التنبؤ مع البيانات بشكل أفضل. إذا كان الثابت c غير موجود ، سيمر الخط من خلال الأصل (0 ، 0) وسيكون لديك تطابق أضعف. ومن ثم ، فإن التحيزات تسمح بتعلم المزيد والمزيد من الاختلافات في الأوزان. باختصار ، تعني الاختلافات الإضافية أن التحيزات تضيف تمثيلًا أكثر ثراءً لمساحة الإدخال إلى الأوزان المستفادة من النموذج. لذلك، يتم حساب المعادلة النهائية للخلايا العصبية على النحو التالي:

$$\text{المخرج} = \sum \left(\text{المدخل} * \text{الوزن}\right) + \text{التحيز}$$

كما ذكرنا سابقًا ، يتم استخدام Perceptron في التصنيف الثنائي. دعنا نفكر في بيرسيبترون بسيط ومع مثال بسيط نفهم بشكل أفضل كيف يعمل في تصنيف البيانات الثنائية. في هذا البيرسبترون لدينا مدخلان x_1 و x_2 مضروبا في الأوزان w_1 و w_2 على التوالي ولدينا أيضًا تحيز:



لنقم أيضًا بإنشاء رسم تخطيطي بمجموعتين مختلفتين من البيانات ممثلة بأشكال دائرية ومستطيلة:



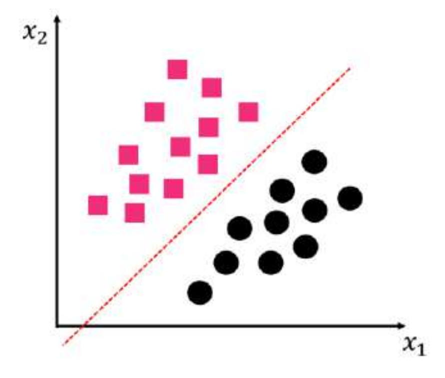
لنفترض أن هدفنا كان فصل هذه البيانات بحيث يكون هناك تمييز بين الدائرة والمستطيل. يمكن لبيرسبترون أن تنشئ حدود قرار لمصنف ثنائي، حيث تكون الحدود هي قرار مناطق المساحة على الرسم البياني الذي يفصل بين نقاط البيانات المختلفة. لفهم هذا بشكل أفضل، دعنا نتعامل مع الدالة قليلاً. يمكننا أن نقول:

$w_1 = 0.5$
$w_2 = 0.5$
$b = 0$

وفقًا لذلك ، ستكون دالة perceptron على النحو التالي:

$$0.5x_1 + 0.5x_2 = 0$$

وسيكون مخططها على النحو التالي:



افترض أن دالة التنشيط، في هذه الحالة، هي دالة خطوية **(step function)** بسيطة تنتج 0 أو 1. تشير دالة perceptron بعد ذلك إلى أشكال المستطيل بمقدار 1 وأشكال الدائرة بمقدار 0. بعبارات أخرى:

$$\begin{cases} 1, & if\ 0.5x_1 + 0.5x_2 \geq 0 \\ 0, & if\ 0.5x_1 + 0.5x_2 < 0 \end{cases}$$

وبالتالي ، فإن الدالة $0.5x_1 + 0.5x_2 = 0$ ى تنشئ حد قرار يفصل بين المستطيلات والدوائر.

خوارزمية التعلم بيرسبترون

تعلم بيرسبترون عملية بسيطة نسبيًا. هدفنا هو الحصول على مجموعة من الأوزان التي تصنف بدقة كل حالة في مجموعة التدريب الخاصة بنا. من أجل تدريب بيرسبترون ، غالبًا ما نقوم بتغذية الشبكة ببيانات التدريب الخاصة بنا عدة مرات. في كل مرة ترى الشبكة مجموعة كاملة من بيانات التدريب ، نقول إن دورة (**epoch**) قد مرت. الدورة هي معامل يحددها المستخدم قبل التدريب.

يمكن تلخيص شبيه الكود لخوارزمية تعلم Perceptron (الخوارزمية 1.3) كالاتي:

يتم "**التعلم**" الحقيقيفي الخطوتين (2.ب) و (2.ج)، أولاً نقوم بتمرير متجه الميزة x_jعبر الشبكة، نحصل على ناتج المنتج الداخلي بأوزان w، ونحصل على الناتج y_j. بعد ذلك، يتم تمرير هذه القيمة للدالة الخطوية التي تُرجع 1 إذا كانت $x > 0$ وإلا 0. الآن نحتاج إلى تحديث متجه الوزن لدينا للتحركفي اتجاه أقرب إلى التصنيف الصحيح للبيانات. تتم إدارة هذه العملية عن طريق **تحديث متجه الوزن بموجب قانون دلتا**في الخطوة (2.ج).

يحدد التعبير $(d_j - y_j)$ ما إذا كان تصنيف المخرجات صحيحًا أم لا. إذا كان التصنيف صحيحًا، فسيكون هذا الاختلاف صفراً. خلاف ذلك، سيكون الاختلاف موجبًا أو سالبًا، مما يمنحنا اتجاهًا تظهر فيه الأوزان (مما يقربنافي النهاية من التصنيف الصحيح). ثم نضرب $(d_j - y_j)$في x_j ، مما يقربنا من التصنيف الصحيح. قيمة α هي **معدل التعلم** (**learning rate**) لدينا وتتحكمفي كبر (أو صغر) الخطوة. من المهم جداً تعيين هذه القيمة بشكل صحيح. على الرغم من أن القيمة الكبيرة لـ α تسمح لنا باتخاذ خطوةفي الاتجاه الصحيح، إلا أنه لا يزال بإمكاننا الوصول إلينا بسهولة من خلال التحسين المحلي أو العالمي.في المقابل، تسمح لنا كمية صغيرة من α باتخاذ خطوات صغيرةفي الاتجاه الصحيح وتضمن أننا لا نتجاوز الحد الأمثل المحلي أو العالمي. ومع ذلك، قد تستغرق هذه الخطوات الصغيرة وقتًا طويلاً حتى يتقارب تعلمنا. أخيرًا، نضيف متجه الوزن السابقفي الوقت t ، $w_j(t)$ ، والذي يكمل عملية الانتقال نحو التصنيف الصحيح. إذا وجدت هذا الطريق التعليمي محيرًا بعض الشيء، فلا تقلق.

يُسمح باستمرار عملية تعلم بيرسيبترون حتى يتم تصنيف جميع عينات التدريب بشكل صحيح أو الوصول إلى عدد محدد مسبقًا (محدد من قبل المستخدم) من الدورات. يتم ضمان الإنهاء إذا كانت α صغيرة بدرجة كافية وكانت بيانات التدريب قابلة للفصل خطيًا. بمعنى آخر، مع الافتراضات المناسبة، يمكن إظهار أن التعلمفي بيرسيبترون سوف يتقارب مع الأوزان

الصحيحة من خلال تكرار الخوارزمية الخاصة به. أي أن تعلم الشبكة سيؤدي إلى تقدير الأوزان التي تمكن الشبكة من إنتاج القيم الصحيحة عند المخرجات.

الخوارزمية 1.3 خوارزمية تعلم بيرسيبترون

1. ابدأ متجه الوزن w بقيم عشوائية صغيرة.
2. حتى يتقارب البيرسيبترون:
   أ. قم بعمل حلقة على كل متجه سمة x_j وقم بتسمية الفئة الفعلية d_j في مجموعة التدريب.
   ب. خذ x وقم بتمريرها عبر الشبكة وحساب قيمة الإخراج:
   $$y_j = f\big(w(t).x_j\big)$$
   ج. قم بتحديث الأوزان w:
   $$w: w_i(t\ +1) =\ w_i(t) + \alpha\big(d_j\ - y_j\big)x_{j,i}$$

السؤال هو، ماذا يحدث إذا كانت بياناتنا غير قابلة للفصل خطيًا أو لدينا اختيار ضعيف في α؟ هل سيستمر التدريب إلى أجل غير مسمى؟ في هذه الحالة ، لا. عادة نتوقف بعد انقضاء عدد معين من الفترات، أو إذا لم يتغير عدد التصنيفات الخاطئة في عدد كبير من الفترات (مما يشير إلى أن البيانات لا يمكن فصلها خطيًا).

تنفيذ بيرسيبترون في بايثون

الآن بعد أن درسنا خوارزمية Perceptron ، دعنا ننفذ الخوارزمية في Python (هذا التطبيق هو فقط لتعريفك بوظيفة perceptron والعملية التعليمية الموجودة في المكتبات ، لذلك إذا بدت صعبة بعض الشيء بالنسبة لك فلا تقلق، لأنه يتم استخدام المكتبات والأطر فقط في المستقبل وليس هناك حاجة لبرمجة هذه العناصر). أدخل الرمز التالي أولاً:

```
# import the necessary packages
import numpy as np

class Perceptron:
  def __init__(self, N, alpha=0.1):
    # initialize the weight matrix and store the learning rate
    self.W = np.random.randn(N + 1) / np.sqrt(N)
    self.alpha = alpha
```

يحدد السطر 5 مُنشئ فئة Perceptron، والذي يقبل معامل مطلوب ثم معامل اختياري:

- N: عدد الأعمدة في متجهات ميزة الإدخال.
- alpha: معدل التعلم لدينا لخوارزمية Perceptron. اضبط هذه القيمة على 0.1 افتراضيًا.

في السطر 7 ، تم أخذ عينات من مصفوفة الوزن W بقيم عشوائية من التوزيع الطبيعي (غاوسي) بمتوسط صفر وتباين الوحدة. تحتوي مصفوفة الوزن على مدخلات N +1 ، واحدة لكل من مدخلات N على متجه الميزة ، بالإضافة إلى إدخال واحد للتحيز. قم بقسمة W على الجذر التربيعي لعدد المدخلات ، وهي تقنية شائعة لقياس مصفوفة الوزن تؤدي إلى تقارب أسرع.

بعد ذلك ، دعنا نحدد الدالة الخطوية:

```
def step(self, x):
  return 1 if x > 0 else 0
```

لتدريب Perceptron، نحدد دالة تسمى fit. إذا كانت لديك خبرة سابقة في التعلم الآلي ومكتبة scikit-Learn ، فأنت تعلم أن تسمية هذه الدالة للتعليم بهذا الاسم أمر شائع:

```
def fit(self, X, y, epochs=10):
  X = np.c_[X, np.ones((X.shape[0]))]
```

تتطلب طريقة fit معاملين إلزاميتن، متبوعة بمعامل اختياري: قيمة X هي بيانات التدريب الخاصة بنا والمتغير y هو تسميات فئة المخرجات (أي ما يجب أن تتنبأ به شبكتنا). أخيرًا، لدينا معامل الدورة التدريبية ، وهي عدد الدورات التي سيقوم Perceptron بتعليمها. يطبق السطر الأخير من الكود التحيز عن طريق إدخال عمود من الوحدات في بيانات التدريب ، مما يسمح لنا بالنظر إلى التحيز كمعامل قابل للتدريب مباشرة داخل مصفوفة الوزن. الآن، دعنا نراجع عملية التدريب الفعلية:

```
    # loop over the desired number of epochs
    for epoch in np.arange(0, epochs):
      # loop over each individual data point
      for (x, target) in zip(X, y):
        # take the dot product between the input features
        # and the weight matrix, then pass this value
        # through the step function to obtain the prediction
        p = self.step(np.dot(x, self.W))
        # only perform a weight update if our prediction
        # does not match the target
        if p != target:
          # determine the error
          error = p - target
          # update the weight matrix
          self.W += -self.alpha * error * x
```

أولاً ، نستخدم حلقة ونشغلها لعدد من الفترات. لكل فترة ، نستخدم أيضًا الإخراج للحلقة في كل نقطة بيانات منفصلة x. بعد ذلك ، يتم أخذ حاصل ضرب الضرب الداخلي بين خصائص الإدخال x ومصفوفة الوزن W لتمرير الإخراج من خلال الدالة الخطوية وللتنبؤ بواسطة بيرسيبترون. سنقوم بتحديث الوزن فقط إذا كانت توقعاتنا لا تتطابق مع الهدف. إذا كان الأمر كذلك ، نحدد الخطأ بحساب العلامة (موجبة أو سالبة).

يتم تحديث مصفوفة الوزن في السطر الأخير من الكود، حيث نتخذ خطوة نحو التصنيف الصحيح. خلال عدد من الدورات ، يمكن لـ Perceptron أن يتعلم الأنماط في البيانات الأساسية ويغير قيم مصفوفة الوزن حتى نتمكن من تصنيف عينات الإدخال لدينا بشكل صحيح.
الدالة الأخيرة التي نحتاج إلى تحديدها هي التنبؤ predict، وكما يوحي الاسم، يتم استخدامها للتنبؤ بتسميات الفئات لمجموعات بيانات الإدخال:

```
  def predict(self, X, addBias=True):
    X = np.atleast_2d(X)
    if addBias:
      X = np.c_[X, np.ones((X.shape[0]))]
    return self.step(np.dot(X, self.W))
```

تتطلب طريقة التنبؤ predict الخاصة بنا مجموعة من بيانات الإدخال X التي يجب تصنيفها. يتم أيضًا إجراء فحص على الكود لمعرفة ما إذا كان يجب إضافة عمود التحيز.
الآن بعد أن تم تنفيذ Perceptron الخاص بنا، دعنا نحاول تطبيقه على مجموعة البيانات البتية (AND و OR و XOR) ونرى كيف يعمل.

نختبرها أولاً في مجموعة البيانات "OR". للقيام بذلك ، أدخل الرمز التالي:

```
# construct the OR dataset
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([[0], [1], [1], [1]])
# define our perceptron and train it
print("[INFO] training perceptron...")
p = Perceptron(X.shape[1], alpha=0.1)
p.fit(X, y, epochs=20)
```

يحدد السطران 2 و 3 مجموعة البيانات "OR". يقوم السطران 6 و 7 بتدريبنا بمعدل تعلم α $= 0.1$ α في 20 دورة تدريبية.

بعد تدريب Perceptron الخاص بنا، نحتاج إلى تقييمه على البيانات للتأكد من أنه قد تعلم بالفعل دالة OR:

```
# now that our perceptron is trained we can evaluate it
print("[INFO] testing perceptron...")
# now that our network is trained, loop over the data points
for (x, target) in zip(X, y):
  # make a prediction on the data point and display the result
  pred = p.predict(x)
  print("[INFO] data={}, ground-truth={}, pred={}".format(
    x, target[0], pred))
```

الكود النهائي على النحو التالي:

```
import numpy as np
class Perceptron:
  def __init__(self, N, alpha=0.1):
```

```
        self.W = np.random.randn(N + 1) / np.sqrt(N)
        self.alpha = alpha

    def step(self, x):
        return 1 if x > 0 else 0

    def fit(self, X, y, epochs=10):
        X = np.c_[X, np.ones((X.shape[0]))]
        for epoch in np.arange(0, epochs):
            for (x, target) in zip(X, y):
                p = self.step(np.dot(x, self.W))
                if p != target:
                    error = p - target
                    self.W += -self.alpha * error * x
    def predict(self, X, addBias=True):
        X = np.atleast_2d(X)
        if addBias:
            X = np.c_[X, np.ones((X.shape[0]))]
        return self.step(np.dot(X, self.W))

X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([[0], [1], [1], [1]])
# define our perceptron and train it
print("[INFO] training perceptron...")
p = Perceptron(X.shape[1], alpha=0.1)
p.fit(X, y, epochs=20)

print("[INFO] testing perceptron...")

for (x, target) in zip(X, y):
    pred = p.predict(x)
    print("[INFO] data={}, ground-truth={}, pred={}".format(
        x, target[0], pred))
```

بعد تنفيذ الكود أعلاه يتم عرض المخرجات على النحو التالي:

```
[INFO] training perceptron...
[INFO] testing perceptron...
[INFO] data=[0 0], ground-truth=0, pred=0
[INFO] data=[0 1], ground-truth=1, pred=1
[INFO] data=[1 0], ground-truth=1, pred=1
[INFO] data=[1 1], ground-truth=1, pred=1
```

كما يتضح، كان Perceptron لدينا قادرًا على تعلم عملية OR. عامل التشغيل OR لـ $x_0 = 0$ و $x_1 = 0$ هو صفر، وجميع المركبات الأخرى متشابهة.

الآن دعنا ننتقل إلى دالة AND، أدخل الكود التالي:

```
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([[0], [0], [0], [1]])
# define our perceptron and train it
print("[INFO] training perceptron...")
p = Perceptron(X.shape[1], alpha=0.1)
p.fit(X, y, epochs=20)
# now that our perceptron is trained we can evaluate it
```

```
print("[INFO] testing perceptron...")
# now that our network is trained, loop over the data points
for (x, target) in zip(X, y):
  # make a prediction on the data point and display the result
  # to our console
  pred = p.predict(x)
  print("[INFO] data={}, ground-truth={}, pred={}".format(
    x, target[0], pred))
```

لاحظ أن سطور التعليمات البرمجية الوحيدة التي تم تغييرها هنا هي السطرين 1 و 2 ، حيث حددنا مجموعة البيانات AND بدلاً من مجموعة البيانات OR .

بعد تنفيذ الكود السابق ، يتم عرض الإخراج على النحو التالي:

```
[INFO] training perceptron...
[INFO] testing perceptron...
[INFO] data=[0 0], ground-truth=0, pred=0
[INFO] data=[0 1], ground-truth=0, pred=0
[INFO] data=[1 0], ground-truth=0, pred=0
[INFO] data=[1 1], ground-truth=1, pred=1
```

وقد لوحظ أنه مرة أخرى كان Perceptron لدينا قادرًا على نمذجة دالة AND بشكل صحيح. يكون عامل التشغيل AND صالحًا فقط عندما يكون كل من $x_0 = 1$ و $x_1 = 1$ ويكون صفرًا لجميع تركيبات AND الأخرى.

أخيرًا ، دعنا نلقي نظرة على وظيفة XOR غير الخطية باستخدام perceptron. أدخل الكود في الأسفل:

```
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([[0], [1], [1], [0]])
# define our perceptron and train it
print("[INFO] training perceptron...")
p = Perceptron(X.shape[1], alpha=0.1)
p.fit(X, y, epochs=20)
# now that our perceptron is trained we can evaluate it
print("[INFO] testing perceptron...")
# now that our network is trained, loop over the data points
for (x, target) in zip(X, y):
  # make a prediction on the data point and display the result
  pred = p.predict(x)
  print("[INFO] data={}, ground-truth={}, pred={}".format(
    x, target[0], pred))
```

من خلال تنفيذ الكود أعلاه، تم الحصول على الإخراج:

```
[INFO] training perceptron...
[INFO] testing perceptron...
[INFO] data=[0 0], ground-truth=0, pred=1
[INFO] data=[0 1], ground-truth=1, pred=1
[INFO] data=[1 0], ground-truth=1, pred=0
[INFO] data=[1 1], ground-truth=0, pred=0
```

لنقم بتشغيل الكود أعلاه مرة أخرى. كان الإخراج هذه المرة على النحو التالي:

```
[INFO] training perceptron...
[INFO] testing perceptron...
[INFO] data=[0 0], ground-truth=0, pred=0
[INFO] data=[0 1], ground-truth=1, pred=0
[INFO] data=[1 0], ground-truth=1, pred=0
[INFO] data=[1 1], ground-truth=0, pred=1
```

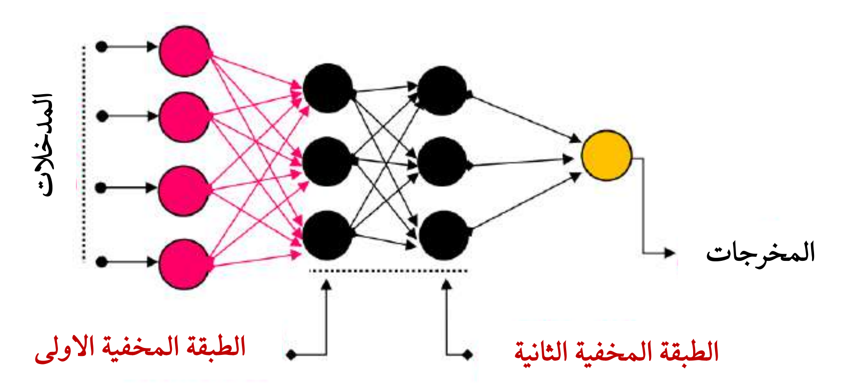
لا يهم عدد المرات التي تجري فيها هذه التجربة بمعدلات تعلم مختلفة أو طرق تهيئة مختلفة ، حيث لا يمكنك أبدًا نمذجة دالة XOR باستخدام Perceptron أحادي الطبقة.

بدلاً من ذلك، ما نحتاجه هو المزيد من الطبقات ذات دوال التنشيط غير الخطية.

Perceptron هو مجرد مصنف خطي ولا يمكنه أبدًا فصل البيانات التي لا يمكن فصلها خطيًا. أيضًا، تُستخدم هذه الخوارزمية فقط لمشاكل التصنيف الثنائية.

بيرسيبترون متعدد الطبقات (الشبكة العصبية امامية التغذية)

كما هو مذكور، فإن التقييد الرئيسي للشبكات العصبية بيرسيبترون هو عدم القدرة على تصنيف البيانات التي لا يمكن فصلها خطيًا. يعد استخدام طبقة مخفيةفي هيكلية الشبكة بمثابة هروب من هذا التقييد. بمعنى آخر، لحل هذا التقييد، يمكن استخدام الطبقة المخفية بين طبقات الإدخال والإخراج. ومن الأمثلة على هذه الشبكات، والتي تعتبر أيضًا أساس التعلم العميق، شبكات بيرسيبترون متعدد الطبقات (**MultilLayer Perceptron**)، أو باختصار **MLP** ، والتي تسمى أيضًا الشبكات العصبية أمامية التغذية (**Feedforward Neural Network**). تعد هذه الشبكات واحدة من أكثر الشبكات استخدامًافي التعلم العميق نظرًا لتوافقها مع مجموعة متنوعة من المشكلات. لأنه لا يوجد حد لإدخاله سواء كانت البيانات صورة أو نص أو فيديو.



الشكل 3ـ2. شبكة عصبية ذات طبقتين مخفيتين

في MLP، تتدفق البيانات في الاتجاه الأمامي من طبقة الإدخال إلى الإخراج. في هذا النوع من الشبكات، بالانتقال من طبقة إلى أخرى، يتم حساب مجموع الاوزان لمجموعة الخلايا العصبية في الطبقة السابقة ونقلها إلى طبقة أخرى عن طريق تطبيق دالة تنشيط غير خطية. سبب تسمية **التغذية الامامية** (**Feed forward**) هو عدم وجود اتصال تغذية مرتدة يتم من خلاله اعتبار مخرجات النموذج مرة أخرى كمدخلات للنموذج (القيم تنتقل فقط من الإدخال إلى الطبقات المخفية ثم إلى الإخراج، ولا توجد قيم يتم إرجاعها إلى الطبقات السابقة، وعلى النقيض من ذلك، تسمح شبكة العودة بإرجاع القيم مرة أخرى). بمعنى آخر، في الشبكة العصبية أمامية التغذية، تتدفق عمليات التنشيط في الشبكة دائمًا إلى الأمام عبر سلسلة من الطبقات. هذه الشبكة هي أيضًا شبكة **متصلة بالكامل** (**Fully Connected**)، لأن كل خلية من الخلايا العصبية للشبكة مترابطة بطريقة تستقبل مدخلات من جميع الخلايا العصبية في الطبقة السابقة وتنقل تنشيط مخرجاتها إلى جميع الخلايا العصبية في الطبقة التالية. يوضح الشكل 3ـ2 مخططًا لشبكة عصبية مع طبقتين مخفيتين.

عندما تقوم الشبكة بمعالجة مجموعة من المدخلات الخارجية، يتم تسليم المدخلات إلى الشبكة من خلال الخلايا العصبية في طبقة الإدخال. هذا يجعل الخلايا العصبية في الطبقة التالية تنتج إشارات تنشيط استجابة لهذه المدخلات. وتتدفق هذه التنشيطات عبر الشبكة لتصل إلى طبقة الإخراج. تنشيط الخلايا العصبية في هذه الطبقة هو استجابة الشبكة للمدخلات والمخرجات النهائية. الطبقات الداخلية للشبكة التي ليست طبقة الإدخال ولا طبقة الإخراج تسمى **الطبقات المخفية**.

عمق الشبكة العصبية يساوي عدد الطبقات المخفية بالإضافة إلى الطبقة الناتجة. لذلك ، تتكون الشبكة في الشكل 3ـ2 من ثلاث طبقات. عدد الطبقات المطلوبة للنظر في عمق الشبكة هو سؤال مفتوح. ومع ذلك، فقد ثبت أن الشبكة التي تحتوي على ثلاث طبقات من الخلايا العصبية (أي طبقتان مخفيتان وطبقة إخراج واحدة) يمكنها تقريب أي دالة إلى الدقة المطلوبة. لذلك ، نحدد هنا الحد الأدنى لعدد الطبقات المخفية المطلوبة لاعتبار الشبكة عميقة طبقتين. بموجب هذا التعريف، يتم وصف الشبكة في الشكل 3ـ2 على أنها شبكة عميقة. ومع ذلك ، تحتوي معظم الشبكات العميقة على أكثر من طبقتين مخفيتين. اليوم ، تحتوي بعض الشبكات العميقة على عشرات أو حتى مئات الطبقات.

تتكون كل شبكة عصبية امامية التغذية من المدخلات والعدد المطلوب من الطبقات المخفية وطبقة تحسب المخرجات تسمى طبقة المخرجات. هذا النهج القائم على الطبقة هو الذي نسميه التعلم العميق، لأن عمق الشبكة العصبية امامية التغذية يصف عدد الطبقات التي تشكل شبكة تغذية عصبية.

يتم تدريب الخلايا العصبية فى MLP باستخدام خوارزمية تعلم الانتشار الخلفى. تعمل MLPs بمثابة مقربات عالمية (**universal approximators**). بمعنى آخر، يمكنهم تقريب أى دالة مستمرة ويمكنهم حل المشكلات التى لا يمكن فصلها خطيًا.

لقد ثبت أن الشبكات العصبية امامية التغذية بطبقة مخفية واحدة فقط يمكن استخدامها لتقريب أى دالة مستمرة.

المشاكل المتعلقة بتصميم وتدريب الشبكات العصبية

الهدف من عملية تعلم الشبكة العصبية هو العثور على مجموعة من قيم الاوزان التي تجعل ناتج الشبكة العصبية مطابقا قدر الإمكان مع القيم المستهدفة الفعلية. هناك العديد من المشكلاتفي تصميم وتدريب شبكة Perceptron متعدد الطبقات:

- **تحديد عدد الطبقات المخفية المطلوب استخدامها في الشبكة.**
- **قرار استخدام خلايا عصبية متعددة في كل طبقة مخفية.** يعد عدد الخلايا العصبية في الطبقة (الطبقات) المخفية أحد أهم القرارات في تصميم الشبكة العصبية. إذا لم يتم استخدام عدد كافٍ من الخلايا العصبية، فلن تكون الشبكة قادرة على نمذجة البيانات المعقدة وسيكون التطابق الناتج ضعيفًا. إذا تم استخدام عدد كبير جدًا من الخلايا العصبية، فقد يكون وقت التدريب طويلًا جدًا ، والأسوأ من ذلك ، قد تؤدي الشبكة إلى الضبط الزائد. عند حدوث الضبط الزائد (Overfitting) ، تبدأ الشبكة في وضع نموذج للتشوهات العشوائيةفي البيانات. والنتيجة هي أن النموذج يتلاءم جيداً مع بيانات التدريب ، ولكنه يعمم بشكل ضعيف على البيانات الجديدة وغير المرئية. يجب استخدام التحقق من الصحة لاختبار هذا.
- **العثور على حل عالمي أمثل يتجنب الحدود الدنيا المحلية.** قد تحتوي الشبكة العصبية النموذجية على مئات الأوزان ، والتي يجب إيجاد قيمها لإنتاج الحل الأمثل. إذا كانت الشبكات العصبية عبارة عن نماذج خطية (مثل الانحدار الخطي) ، فإن العثور على مجموعة الأوزان المثلى ليس بالأمر الصعب. لكن ناتج الشبكة العصبية كدالة للمدخلات غالبًا ما يكون غير خطي للغاية. هذا يعقد عملية التحسين. إذا قمت برسم الخطأ كدالة للأوزان ، فمن المحتمل أن ترى سطحًا غير مستوٍ به العديد من الحدود الدنيا المحلية على النحو التالي:



هذه الصورة بسيطة للغاية لأنها تعرض فقط قيمة وزن واحد (على المحور الأفقي).

- **التقارب إلى الحل الأمثل في فترة زمنية معقولة.**
- **التحقق من صحة الشبكة العصبية لاختبار الضبط الزائد.**

دالة التنشيط

دالة التنشيط مهمة جداًفي التعلم العميق. الغرض من دوال التنشيط هو الحصول على رقم من الإدخال وحساب سلسلة من العمليات الحسابية لإنتاج ناتج بين 0 و 1 أو ـ1 إلى 1. وظيفة التنشيطفي كل خلية عصبية اصطناعية، إذا وصلت الإشارات المستلمة إلى الحد الأدنى، فإنها ترسل إشارات الاخراج إلى المستوى التالي. ببساطة، تقرر دالة التنشيط ما إذا كان يجب تنشيط الخلية العصبية ام لا.

في التعلم العميق، الشبكة العصبية بدون دالة التنشيط هي مجرد نموذج انحدار خطي بسيط!؟ لأن هذه الدوال تؤديفي الواقع حسابات غير خطية عند مدخلات الشبكة العصبية، مما يمكنها من التعلم وتنفيذ مهام أكثر تعقيداً. لذلك، تعد دراسة الأنواع المختلفة وتحليل مزايا وعيوب كل دالة تنشيط ضرورية لتحديد النوع المناسب لدالة التنشيط التي يمكن أن توفر عدم الخطية والدقة في نموذج شبكة عصبية معين. أيضًا، بسبب مشكلة تلاشي الانحدار الاشتقاقي (**Vanishing gradient**)، التي سنتحدث عنها لاحقًا، يعد إعداد دالة التنشيط الصحيحة للشبكة أمرًا مهمًا للغاية.

لا يمكن أن تكون دالة التنشيط المستخدمةفي الشبكات العميقة أي دالة، ولكن يجب أن يكون لها خصائص معينة. من السمات المهمة لدالة التنشيط أنه يجب أن تكون مشتقة. تتعلم الشبكة من الأخطاء المحسوبةفي طبقة الإخراج. تؤدي دالة تنشيط مشتقة لتحسين الانتشار الخلفي (**propagating backwards**) لحساب تدرجات الخطأ فيما يتعلق بالأوزان ثم تحسين الأوزان باستخدام الانحدار الاشتقاقي (أو أي تقنية تحسين أخرى لتقليل الأخطاء).

يتمثل دور دالة التنشيط فى إخراج مجموعة من قيم الإدخال المعطاة إلى خلية عصبية (أو طبقة).

لماذا دوال التنشيط غير الخطية ضرورية؟

تقدم دالة التنشيط خطوة إضافيةفي كل طبقة أثناء **الانتشار الأمامي (forward propagation)**، لكن الأمر يستحق الحساب. لنفترض أن لدينا شبكة عصبية تعمل بدون دوال التنشيط.في هذه الحالة، تقوم كل خلية عصبية بتحويل خطي على المدخلات باستخدام الأوزان والتحيزات فقط. لذلك لم يعد من المهم استخدام طبقات مخفية متعددةفي شبكتك العصبية لأن جميع الطبقات تتصرف بالطريقة نفسها لأن الجمع بين دالتين الخطيتين هو دالة خطية ولن تتمتع الشبكة بقوة أكبر من الانحدار الخطي.

افترض أن لديك متجه إدخال x وثلاث طبقات مخفية ممثلة بمصفوفات الوزن W_1، W_2 و W_3. بدون أي دالة تنشيط، تحتوي شبكتك العصبية على الناتج y=x $W_1W_2W_3$ الذي يساوي y=xW بحيث يكون $W = W_1W_2W_3$ وهذا ليس سوى مضاعفة المصفوفة. الآن، مع إدخال دالة التنشيط غير الخطية بعد كل تحويل خطي، لم يعد هذا يحدث:

$$y = f_1\left(W_1 f_2(W_2 f_3(W_3 x))\right)$$

يمكن الآن إنشاء كل طبقة فوق نتائج الطبقة غير الخطية السابقة، والتي ينتج عنها أساسًا دالة معقدة غير خطية.

تمت إضافة دالة تنشيط إلى الشبكة العصبية الاصطناعية لمساعدة الشبكة على تعلم الأنماط المعقدة في البيانات.

الميزات المثلى لدالة التنشيط

كما ذكرنا سابقًا، يجب أن يكون لدالة التنشيطفي الشبكات العصبية خصائص معينة. يمكن تلخيص الميزات المرغوبة التي يجب أن تتوفرفي دالة التنشيط على النحو التالي:

- **غيرخطى (Nonlinear)**: كما ذكرنا سابقًا، إذا كانت دالة تنشيط خطية ، فيمكن بسهولة البيرسيبترون الذي يحتوي على عدة طبقات مخفية في Perceptron أحادي الطبقة ، لأنه يمكن ببساطة التعبير عن التركيب الخطي لمتجه الإدخال كتكوين خطي واحد لمتجه الإدخال . في هذه الحالة ، لن يتأثر عمق الشبكة. لذلك ، فإن اللاخطية في دالة التنشيط ضرورية عندما تكون حدود القرار غير خطية بطبيعتها. نظرًا لأن الشبكة العصبية الاصطناعية تتعلم الأنماط أو الحدود من البيانات ، فإن اللاخطية في دالة التنشيط ضرورية حتى يمكن للشبكة العصبية الاصطناعية أن تتعلم بسهولة أي حدود خطية أو غير خطية.
- **المركزـ الصفري (Zero-Centered)**: يجب أن يكون ناتج دالة التنشيط مركزًا صفريًا حتى لا تتغير التدرجات في اتجاه معين. يتم استدعاء دالة المركز الصفري عندما يكون لمداها قيم موجبة وسالبة. إذا لم تكن دالة تنشيط شبكة مركزية صفرية ، فهي دائمًا إيجابية أو سلبية دائمًا. لذلك ، يتم دائمًا تمرير ناتج الطبقة إلى قيم موجبة أو سالبة. نتيجة لذلك، يحتاج متجه الوزن إلى مزيد من التحديث ليتم تدريبه بشكل صحيح. لذلك ، إذا لم تكن دالة التنشيط مركزًا صفريًا ، فسيزداد عدد الدورات التدريبية المطلوبة لتدريب الشبكة. هذا هو سبب أهمية ميزة المركز الصفري ، وإن لم تكن ضرورية.
- **التكلفة الحسابية (Computational Expense)**: يتم تطبيق دوال التنشيط بعد كل طبقة ويجب حسابها ملايين المرات في الشبكات العميقة. لذلك ، يجب أن يكون حسابهم قليلا من الناحية الحسابية.

- **قابل للاشتقاق (Differentiable):** يتم تدريب الشبكات العصبية باستخدام عملية الانحدار الاشتقاقي، لذلك يجب اشتقاق دالة التنشيط وفقًا للإدخال. هذا مطلب أساسي لتشغيل دالة التنشيط.
- **مستمر (Continuous):** يمكن اشتقاق دالة ما لم تكن مستمرة.
- **محاط(Bounded):** يتم إرسال بيانات الإدخال من خلال سلسلة من بيرسيبترون، كل منها يحتوي على دالة التنشيط. نتيجة لذلك ، إذا لم تقتصر الدالة على حد واحد ، فقد تنفجر قيمة الإخراج (explode). للسيطرة على هذا الانفجار في القيم ، فإن الطبيعة المحدودة لدالة التنشيط مهمة ولكنها ليست ضرورية.

مشاكل دوال التنشيط

تعد مشكلة تلاشي الاشتقاق **(Vanishing Gradient problem)** ومشكلة الخلايا العصبية الميتة **(dead neuron)** مشكلتين رئيسيتين تواجههما وظائف التنشيط المستخدمة على نطاق واسع:

- **مشكلة تلاشي الاشتقاق:** يتم تدريب الشبكات العصبية باستخدام الانحدار الاشتقاقي وخوارزمية الانتشار الخلفي. عند استخدام خوارزمية الانتشار الخلفي ، يتم حساب التدرج بشكل أصغر وأصغر في المرحلة العكسية. هذا لأنه يجد الانحدار الاشتقاقي في كل تكرار للمشتقات الجزئية بالمرور من الطبقة الأخيرة إلى الطبقة الأولية باستخدام قانون السلسلة. في شبكة بها n طبقات مخفية ، يتم ضرب مشتقات هذه الطبقة n ببعضها البعض. الآن إذا كانت هذه المشتقات صغيرة ، فإن الانتقال إلى الطبقات الأولية سينخفض بشكل كبير (أو في أسوأ الحالات ستكون صفراً وسيتوقف تعلم الشبكة) سيؤدي ذلك إلى ظاهرة **تلاشي التدرج (Vanishing Gradient)**او تلاشي الاشتقاق. نظرًا لأن هذه التدرجات الصغيرة لا يتم تحديثها في تكرار الخوارزمية ، وغالبًا ما تكون هذه الطبقات الأولية فعالة في التعرف على البيانات ، فإنها تؤدي إلى عدم دقة الشبكة بشكل غير كاف ولا يمكن لهذه الطبقات التعلم بشكل صحيح. بمعنى آخر ، تختفي تدرجاتها بسبب عمق الشبكة وتنشيطها ، مما يقلل القيمة إلى الصفر. وبالتالي ، لا نريد تغيير دالة تنشيط التدرج إلى الصفر. على عكس تلاشي الانحدار ، هناك **انحدار متفجر(Exploding Gradients)**. إذا كانت قيم التدرجات كبيرة ، فإنها تؤدي إلى تحديثات كبيرة جداً لوزن نموذج الشبكة العصبية أثناء التدريب. مع النمو الأسي من خلال النقل إلى الطبقات ، تتدفق هذه التدرجات الكبيرة في النهاية ولن تكون الأوزان قادرة على التحديث ، مما يؤدي إلى إنشاء شبكات غير مستقرة.

- **الخلايا العصبية الميتة:** عندما تفرض دالة التنشيط جزءًا كبيرًا من المدخلات على الصفر أو بالقرب من الصفر ، فإن تلك الخلايا العصبية المقابلة تكون غير نشطة (ميتة) لمساعدة الناتج النهائي. أثناء تحديث الأوزان ، من الممكن تحديث الأوزان بطريقة تجعل الوزن الإجمالي لجزء كبير من الشبكة صفريًا بالقوة. بالكاد يمكن للشبكة أن تتعافى من مثل هذا الموقف ، ولا يمكن لجزء كبير من المدخلات أن يساعد الشبكة. هذا يؤدي إلى مشكلة لأنه قد يتم تعطيل جزء كبير من الإدخال تمامًا أثناء تنفيذ الشبكة. تسمى هذه الخلايا العصبية التي تصبح خاملة بشدة "الخلايا العصبية الميتة" **(dead neuron)** وتسمى هذه المشكلة مشكلة الخلايا العصبية الميتة. باختصار ، الخلايا العصبية الميتة هي شبكة عصبية اصطناعية تسمى الخلايا العصبية التي لا يتم تنشيطها أثناء التدريب. يمنع هذا العصبون من التعبير عن وزنه لأن مشتقاته ستكون صغيرة جداً أو صفرية. لا تنتشر الأخطاء عبر الخلايا العصبية الميتة ، لذا فهي تؤثر على الخلايا العصبية الأخرى في الشبكة.

دوال التنشيط المفيدة

في هذا القسم، سنناقش وظائف التنشيط الأكثر استخدامًا وخصائصها.

Sigmoid

يتم تعريف وظيفة التنشيط Sigmoid على النحو التالي:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

حيث x هي مدخلات دالة المنشط. الآن دعنا نبرمج هذافي بايثون:

```
def sigmoid(x):
  return 1/(1+np.exp(-x))
```

دالة Sigmoid متصلة ومحصورةفي النطاق (0،1) ومشتقة ولكن ليست صفرية المركز. تأخذ هذه الوظيفة أي قيمة حقيقية كمدخلات وترجع القيمفي النطاق من 0 إلى 1في المخرجات. كلما كان المدخل أكبر (أكثر إيجابية)، كلما كان الناتج أقرب إلى 1.0 ، بينما كلما كان المدخل أصغر (أكثر سلبية)، كلما كان الناتج أقرب إلى 0.0.

مشتق دالة Sigmoid هو $\sigma(x)$، دالة Sigmoid $\sigma(x)$، مضروبةفي $1-\sigma(x)$:

$$\acute{\sigma}(x) = \sigma(x).(1-\sigma(x))$$

والتيفي بايثون يمكننا أن نبرمج Sigmoid على النحو التالي:

```
def der_sigmoid(x):
  return sigmoid(x) * (1- sigmoid(x))
```

الآن دعنا نوضح دالة Sigmoid ومشتقتها. للقيام بذلك ، أدخل الكود التالي:

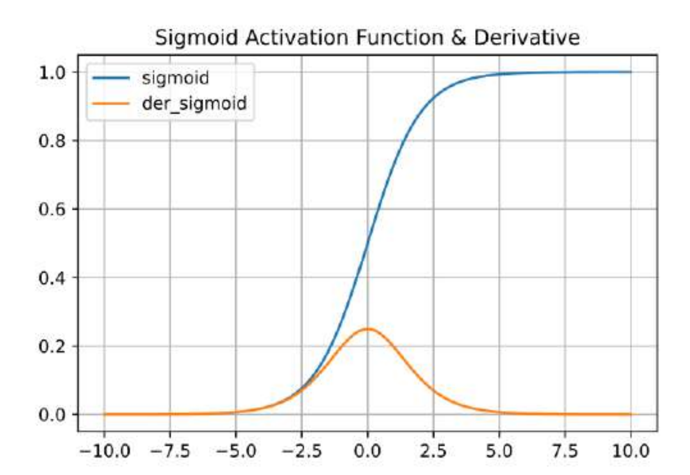
```
import numpy as np
import matplotlib.pyplot as plt

# Sigmoid Activation Function
def sigmoid(x):
  return 1/(1+np.exp(-x))

# Derivative of Sigmoid
def der_sigmoid(x):
  return sigmoid(x) * (1- sigmoid(x))

# Generating data to plot
x_data = np.linspace(-10,10,100)
y_data = sigmoid(x_data)
dy_data = der_sigmoid(x_data)

# Plotting
plt.plot(x_data, y_data, x_data, dy_data)
plt.title('Sigmoid Activation Function & Derivative')
plt.legend(['sigmoid','der_sigmoid'])
plt.grid()
plt.show()
```



تعاني دالة Sigmoid من بعض العيوب الرئيسية. تقتصر دالة Sigmoid على النطاق (0،1). ومن ثم فإنه ينتج دائمًا قيمة غير سلبية. لذلك فهي ليست دالة تنشيط صفرية المركز. تحول دالة Sigmoid نطاقًا كبيرًا من المدخلات إلى نطاق صغير (0،1). لذلك يؤدي التغيير الكبير في قيمة الإدخال إلى تغيير طفيف في قيمة الإخراج. يؤدي هذا أيضًا إلى قيم متدرجة صغيرة. بسبب قيم التدرج المنخفضة ، يتلاشى الانحدار.

في التطبيقات العملية، فإن دالة Sigmoid، على الرغم من شعبيتها في الماضي، أصبحت أقل استخدامًا بسبب مشكلتين مهمتين:

1. **مشكلة تلاشي الانحدار.** تتسبب دالة Sigmoid في مشكلة تلاشي الانحدار في خوارزمية الانتشار الخلفي. في هذه الحالة ، لا يتم إرسال أي إشارة عبر الخلايا العصبية ، لذلك لن تتعلم الخلايا العصبية أي شيء أثناء مرحلة التدريب.
2. **ليست صفرية المركز.** مخرجات دالة Sigmoid ليست صفرية المركز. ومن ثم، في خوارزمية الانتشار الخلفي ، يتم إنشاء التدرجات التي تكون إما إيجابية أو كلها سلبية، وهي غير مناسبة لتحديث تدرجات الأوزان.

تُستخدم دالة Sigmoid عمومًا في مشاكل التصنيف الثنائي وتصنيف العلامات المتعددة في طبقة المخرجات لأنه يتعين علينا توقع الاحتمال كناتج. نظرًا لأنه من المحتمل أن يكون أي شيء بين النطاقين 0 و 1 ، فإن Sigmoid يعد اختيارًا جيدًا بسبب نطاقه.

المزايا

- يتراوح نطاق إخراجها من 0 إلى 1 ، لذلك يمكنها إنشاء الاحتمالات. هذا يجعل Sigmoid مفيداً للخلايا العصبية الناتجة عن الشبكة العصبية لأغراض التصنيف.
- إنها قابلة للاشتقاق في كل مكان.
- طبيعتها غير خطية.

العيوب

- يعاني من مشكلة التشبع (saturation problem). تعتبر الخلية العصبية مشبعة إذا وصلت إلى الحد الأقصى أو الحد الأدنى لقيمتها ، بحيث يكون مشتقها يساوي 0. في هذه الحالة ، لا يتم تحديث الأوزان ، مما يؤدي إلى ضعف التعلم للشبكات العميقة.
- انها ليست دالة صفرية المركز. وبالتالي ، فإن تدرج جميع الأوزان المرتبطة بالخلايا العصبية يكون موجبًا أو سلبيًا. أثناء عملية التحديث ، يُسمح لهذه الأوزان بالتحرك في اتجاه واحد فقط ، أي إيجابية أو سلبية في كل مرة. هذا يجعل الأمر أكثر صعوبة لتحسين دالة الخطأ.
- لها تكلفة حسابية عالية. تؤدي هذه الدالة عملية أسية ينتج عنها وقت حسابي أكبر.

tanh

ترتبط دالة تنشيط tanh ارتباطًا وثيقًا بدالة Sigmoid وشكلها الرياضي هو كما يلي:

$$f(x) = \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = 2\sigma(2x) - 1.$$

في بايثون يمكننا برمجتها على النحو التالي:

```
def htan(x):
  return (np.exp(x) - np.exp(-x))/(np.exp(x) + np.exp(-x))
```

كما يتضح من المعادلة أعلاه، فإن tanh هو ببساطة نسخة مصغرة من Sigmoid. ومع ذلك، فهو صفري المركز. وبالتالي، فإنه لا يظهر بعض المشاكل التي يعاني منها Sigmoid Activator. هذه الدالة مستمرة ومشتقة ومحصورة في النطاق (ـ1،1). وبالتالي، فإنه ينتج سالب، موجب، وصفر كمخرجات، ويتم تعيين المدخلات السلبية بشدة إلى tanh كمخرجات سالبة. لذلك، فإن دالة التنشيط tanh تتمحور حول الصفر وتحل مشكلة صفر المركز لدالة Sigmoid.

يتم حساب مشتق دالة tanh على النحو التالي:

$$f(x) = 1 - f(x)^2$$

والتي في بايثون يمكننا كتابتها على النحو التالي:

```
def der_htan(x):
  return 1 - htan(x) * htan(x)
```

دعنا نوضح دالة tanh ومشتقاتها. للقيام بذلك ، أدخل الكود التالي:

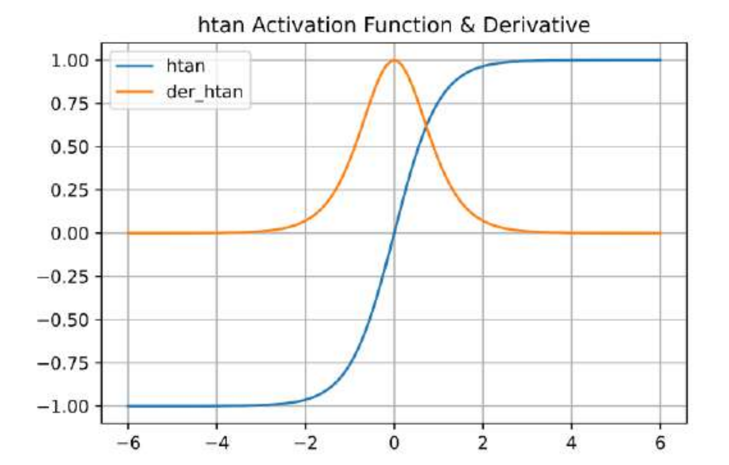
```
import numpy as np
import matplotlib.pyplot as plt

# Hyperbolic Tangent (htan) Activation Function
def htan(x):
  return (np.exp(x) - np.exp(-x))/(np.exp(x) + np.exp(-x))

# htan derivative
def der_htan(x):
  return 1 - htan(x) * htan(x)

# Generating data for Graph
x_data = np.linspace(-6,6,100)
y_data = htan(x_data)
dy_data = der_htan(x_data)

# Graph
plt.plot(x_data, y_data, x_data, dy_data)
plt.title('htan Activation Function & Derivative')
plt.legend(['htan','der_htan'])
plt.grid()
```



المزايا

- على عكس Sigmoid، فهي دالة صفرية المركز لتسهيل تحسين دالة الخطأ.
- تنعيم ناتج الخلايا العصبية في النطاق بين 1 و 1.

العيوب

- انها مكلفة حسابيا.
- إنها عرضة لتلاشي الانحدار.

ReLU

لا يمكن استخدام دوال Sigmoid و tanh في الشبكات متعددة الطبقات بسبب مشكلة تلاشي الانحدار. دالة التنشيط ReLU ، وهي دالة التنشيط الأكثر شيوعًا في التعلم العميق (الطبقة المخفية) ، تتغلب على مشكلة تلاشي الانحدار، مما يسمح للشبكة بالتعلم بشكل أسرع وأداء أفضل. يتم تعريف دالة ReLU على النحو التالي:

$$f(x) = \begin{cases} 0 \ for \ x \leq 0 \\ x \ for \ x > 0 \end{cases}$$

في بايثون يمكننا كتابتها على النحو التالي:

```
def ReLU(x):
  data = [max(0,value) for value in x]
  return np.array(data, dtype=float)
```

يتم حساب مشتق دالة ReLU على النحو التالي:

$$f́(x) = \begin{cases} 0 \; for \; x \leq 0 \\ 1 \; for \; x > 0 \end{cases}$$

في بايثون يمكننا كتابتها على النحو التالي:

```
def der_ReLU(x):
  data = [1 if value>0 else 0 for value in x]
  return np.array(data, dtype=float)
```

أدخل الكود التالي لتوضيح دالة ReLU ومشتقاتها:

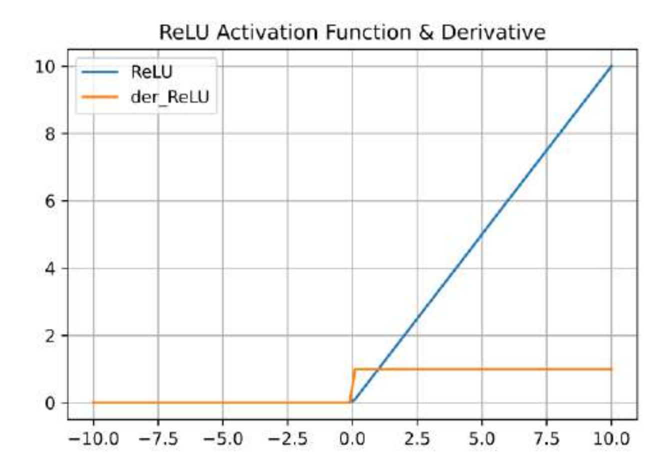
```
import numpy as np
import matplotlib.pyplot as plt

# Rectified Linear Unit (ReLU)
def ReLU(x):
  data = [max(0,value) for value in x]
  return np.array(data, dtype=float)

# Derivative for ReLU
def der_ReLU(x):
  data = [1 if value>0 else 0 for value in x]
  return np.array(data, dtype=float)

# Generating data for Graph
x_data = np.linspace(-10,10,100)
y_data = ReLU(x_data)
dy_data = der_ReLU(x_data)

# Graph
plt.plot(x_data, y_data, x_data, dy_data)
plt.title('ReLU Activation Function & Derivative')
plt.legend(['ReLU','der_ReLU'])
plt.grid()
plt.show()
```



المزايا

- يسرع بشكل كبير تقارب الانحدار الاشتقاقي العشوائي مقارنة بدالة Sigmoid.
- يمكنه التعامل مع مشكلة تلاشي الانحدار.
- إنها دالة حسابية رخيصة.
- إنها دالة التنشيط الأكثر استخداماً.
- يتقارب بسرعة كبيرة.

العيوب

- ليست صفرية المركز.
- لديه مشكلة مع الخلايا العصبية الميتة. الجانب السلبي من الرسم البياني يعيد ضبط قيمة التدرج إلى الصفر. لهذا السبب، لا تحدث الأوزان والتحيزات لبعض الخلايا العصبية أثناء عملية الانتشار الخلفي. هذا يمكن أن ينتج خلايا عصبية ميتة لا يتم تنشيطها أبداً. بمعنى آخر، تكون جميع قيم الإدخال السلبية صفرية على الفور، مما يقلل من قدرة النموذج على تعليم البيانات بشكل صحيح.

عندما يتلقى ReLU مدخلات سلبية، يكون الناتج صفراً. لذلك، لا يتعلم أي شيء من خلال النشر اللاحق (لأنه لا يمكنك عكسه). بمعنى آخر، إذا كان المشتق صفراً، فإن التنشيط الإجمالي يساوي صفراً، لذلك لا توجد مساهمة لتلك الخلايا العصبية في الشبكة.

Softmax

تُستخدم دالة sofmax كمخرج في مسائل التصنيف متعدد الفئات لإيجاد احتمالات لفئات مختلفة (على عكس Sigmoid، الذي يُفضل للتصنيف الثنائي). تحسب وظيفة Softmax احتمالات كل فئة هدف على جميع الفئات المستهدفة الممكنة (مما يساعدفي تحديد الفئة المستهدفة):

$$Softmax\ (z_i) = \frac{e^{z_i}}{\sum_{k=1}^{k} e^{z_k}}\ for\ j = 1, \dots, k$$

تُرجع Softmax الاحتمال لنقطة بيانات تنتمي إلى كل فئة على حدة. لاحظ أن مجموع كل القيم هو 1.

في بايثون يمكننا برمجتها على النحو التالي:

```
def softmax(x):
    return np.exp(x) / np.sum(np.exp(x), axis=0)
```

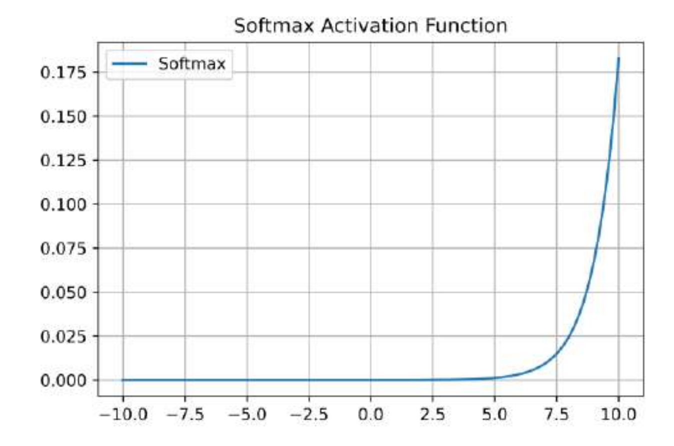
أدخل الكود التالي لتوضيح دالة Softmax:

```
import numpy as np
import matplotlib.pyplot as plt

# Softmax Activation Function
def softmax(x):
    return np.exp(x) / np.sum(np.exp(x), axis=0)

# Generating data to plot
x_data = np.linspace(-10,10,100)
y_data = softmax(x_data)

# Plotting
plt.plot(x_data, y_data)
plt.title('Softmax Activation Function')
plt.legend(['Softmax'])
plt.grid()
plt.show()
```



بالنسبة لمشاكل التصنيف متعدد الطبقات، تكون طبقة المخرجات كبيرة مثل الفئة المستهدفة من الخلايا العصبية. على سبيل المثال، افترض أن لديك 4 فئات [A, B, C, D]. ومن ثم سيكون هناك 4 خلايا عصبيةفي طبقة الإخراج. لنفترض أن ناتج دالة Softmax لبيانات ما يساوي [0.26،0.14،0.41،0.19]، بالنظر إلى قيمة الاحتمال يمكننا القول إن الإدخال ينتمي إلى الفئة C.

التحسين ودالة الخطأ (Loss Function)

خوارزميات التحسين عبارة عن خوارزميات تُستخدم لتقليل دالة الخطأ / التكلفة (دالة الخطأ هي خطأ عينة تعليمي واحد، بينما دالة التكلفة هي خطأ مجموعة البيانات التعليمية بأكملها) عن طريق تحديث أوزان الشبكة. تلعب خوارزميات التحسين دورًا مهمًا للغايةفي عملية تدريب الشبكة ولها تأثير مباشر على وقت التدريب الذي يقضيه. بمعنى آخر، خوارزميات التحسين للتعلم العميق هي المسؤولة عن العمل المعقد لنماذج التعلم العميق للتعلم من البيانات.

لفهم ما هو التحسين؟ يجب علينا أولا تحديد الهدف. يعتمد الهدف على ميزات معينة للنظام تسمى المتغيرات. هدفنا هو إيجاد قيم المتغيرات التي تعمل على تحسين الهدف. معظم المتغيرات محدودة نوعًا ما. من وجهة نظر رياضية، التحسين هو عملية تكبير (**maximizing**) أو تصغير (**minimizing**) دالة موضوعية $f(x)$ من خلال البحث عن متغيرات مناسبة x فيما يتعلق بقيود c_i، والتي يمكن ضغطها على النحو التالي:

$$min_{x\epsilon R^n} f(x) subject\ to \begin{cases} c_i(x) = 0, & i\ \epsilon\ \mathcal{E} \\ c_i(x) \geq 0, & i\ \epsilon\ \mathcal{I} \end{cases}$$

حيث $\mathcal{E}$ و $\mathcal{I}$ هما على التوالي مجموعة من المؤشرات لقيود المساواة وعدم المساواة. هذه العبارة الحسابية مخيفة بالتأكيد للوهلة الأولى !!، ربما لأنها محسّنة لوصف عام. لكن لا تقلق، فكل شيء سيتضح لاحقًا.

هناك عدة طرق مختلفة لتحسين التعلم العميق. على سبيل المثال، أبسط خوارزمية تحسين مستخدمةفي التعلم العميق هي **الانحدار الاشتقاقي Gradient Descent**. الانحدار الاشتقاقي هو خوارزمية تحسين تكرارية من الدرجة الأولى تستخدم للعثور على الحد الأدنى المحلي لدالة معينة. هذا يعني أنه يتم أخذ المشتق الأول فقطفي الاعتبار عند تحديث المعاملات. في كل تكرار، نقوم بتحديث المعاملاتفي الاتجاه المعاكس لتدرج الوظيفة الهدف $J(\theta)$. يتم تحديد حجم الخطوة التي نتخذهافي كل تكرار للوصول إلى الحد الأدنى المحلي من خلال معدل التعلم α. لذلك، نتبع اتجاه التدرج لأسفل للوصول إلى الحد الأدنى المحلي.

طريقة أخرى هي تحسين نيوتن، والتي تساعد على تحسين المحسنات باستخدام مشتق من الدرجة الثانية من خلال إيجاد جذور الدالة. ومع ذلك، فإن هذه الطريقة تزيد بشكل كبير من التعقيد الحسابي مقارنة بطرق الانحدار الاشتقاقي، والتي تعتمد على مشتقة من الدرجة الأولى. لذلك، يُفضل تقليل التدرجاتفي تدريب الشبكة العصبية. هناك خوارزميات مختلفة تعتمد على الانحدار الاشتقاقي. استمرارًا لهذا القسم، وبعد وصفه الكامل، سنقدم ونراجع إصدارات أخرى من هذه الخوارزمية.

التعريف 1.3 المحسنات

في عملية تدريب الشبكة العصبية، يبحث المحسنات عن مجموعة من المعاملات، تسمى أيضًا الأوزان، لتقليل مقدار الخطأ الذي يلحق بدالة الخطأ.

الانحدار الاشتقاقي[1]

تعد خوارزمية الانحدار الاشتقاقي طريقة تحسين قائمة على التكرار تحاول تقليل مقدار دالة الخطأ عن طريق تغيير الأوزان الداخلية للشبكة العصبية.في هذه الطريقة، يتم تحديث أوزان

[1] Gradient Descent

الشبكة تدريجيًا وفي كل تكرار، تحاول الخوارزمية تقليل قيمة دالة الخطأ. يتم تنفيذ هذه العملية طالما أن تكرارها لا يؤدي إلى تغييرفي دالة الخطأ.

هناك ثلاث طرق عامة لاستخدام الانحدار الاشتقاقي: الانحدار الاشتقاقي الكامل **(Batch gradient descent)**، الانحدار الاشتقاقي دفعة صغيرة **(Mini-batch gradient descent)**. عندما يتم التحديث بعينة واحدة فقط، يُطلق عليه طريقة **الانحدار الاشتقاقي العشوائي** (**Stochastic gradient descent**) او اختصارا (**SGD**).

في هذه طريقة الانحدار الاشتقاقي العشوائي، يتم الحصول على إجراءات وأوزان جديدة عن طريق إدخال كل عينةفي شبكة التحديث. عيب هذه الطريقة أنها تتعثرفي الحد الأدنى المحلي. بالإضافة إلى ذلك، فإنه يحتوي على نتائج غير مستقرة بسبب الاستجابة لكل إدخال عينةفي الشبكة.

يقلل الانحدار الاشتقاقى العشوائى من وقت التحديث ويزيل بعض التكرار الحسابى ، مما يؤدى إلى تسريع العمليات الحسابية بشكل كبير.

في طريقة الانحدار الاشتقاقي الكامل يتم تغذية الشبكة بجميع عينات التدريب. تقوم الشبكة بإجراء تحديث مرة واحدة لجميع العينات بعد حساب الخطأ. على الرغم من أن هذه الطريقة تساعد النموذجفي الهروب من الحد الأدنى المحلي وتوفر تقاربًا أكثر ثباتًا مقارنةً بالانحدار الاشتقاقي للعينة، إلا أنها مع ذلك تؤدي إلى وقت تدريب أطول. أيضًا، ليس من الممكن دائمًا إرسال جميع عينات التدريب إلى الشبكة بسبب نقص الذاكرة.في الفجوة بين الطريقتين، يتم استخدام الانحدار الاشتقاقي دفعة صغيرة (**mini-batch gradient descent**).في هذه الطريقة، يتم تغذية الشبكة بمجموعة من عينات التدريب للاستفادة من الطريقتين السابقتين. يتمتع تحديث المعاملات باستخدام مجموعة من العينات بميزة مهمة: باستخدام هذه الطريقة، يكون النموذج أكثر مقاومة من العينات الصاخبة (Noisy data)ولديه تباين أقلفي تحديث المعاملات. هذا يوفر تقاربا أكثر استقرارا. ومع ذلك، تتطلب هذه الأساليب اختيار معدل التعلم (α) الذي ليس من السهل دائمًا اختياره. بالإضافة إلى ذلك، لا يمكن أن يكون معدل التعلم نفسه في جميع المراحل التدريبية ولجميع المعايير المختلفة هو الأمثل. إذا تم اختيار α بحجم كبير جدًا، فقد يتقلب التدريب أو لا يتقارب أو يتجاوز الحدود الدنيا المحلية ذات الصلة.في المقابل، إذا تم اختيار معدل تعليمي صغير جدًا، فإنه يؤخر بشكل كبير عملية التقارب. ومن ثم، فإن الأسلوب الشائع للتحايل على هذا هو استخدام **انحلال معدل التعلم** (**rate decay**). على سبيل المثال، يمكن أن يؤدي استخدام الانحلال إلى تقليل معدل التعلم بعدة فترات. هذا يسمح بقدر أكبر من التعلمفي بداية التدريب ومعدل تعلم أقلفي نهاية التدريب. ومع ذلك، فإن طريقة الانحلال هذه هي أيضًا معاملات فائقةفي حد ذاتها ويجب تصميمها بعناية اعتمادًا على التطبيق.

الهدف من مُحسِّن معدل التعلم التكيفي هو حل مشكلة إيجاد معدل التعلم الصحيح.في هذه الأساليب، لا يكون معدل التعلم α متغيرًا شاملاً، ولكن بدلاً من ذلك يكون لكل معامل قابل للتعليم معدل التعلم الخاص بها.في حين أن هذه الطرق لا تزال بحاجة إلى معاملات، الا انها تعمل بشكل جيد مع نطاق أوسع من التنظيمات؛في كثير من الأحيان، انهم فقط يستخدمون المعاملات الفائقة الافتراضية المقترحة.

تنفيذ الانحدار الاشتقاقي في بايثون

كما ذكرنا سابقًا،في التحسين أردنا تصغير دالة. لنرى الآن كيف يمكننا حل دالة min؟ بفضل حساب التفاضل والتكامل، لدينا أداة تسمى الانحدار. تخيل كرة على قمة تل. نحن نعلم أن التلال لها تدرجات مختلفةفي أماكن مختلفة. بسبب الجاذبية، تتحرك الكرة أسفل منحنى التل.في أي اتجاه تذهب الكرة؟ منحدر شديد الانحدار. بعد فترة، تصل الكرة إلى الحد الأدنى المحلي حيث تكون الأرض مسطحة بالنسبة لمحيطها. هذه هي طبيعة الانحدار الاشتقاقي. يمكننا تحويل المسار السلس للكرة إلى خطوات صغيرة.في الخطوة k ، سيكون لدينا كميتان: طول الخطوة α_k والاتجاه p_k. لمشاهدة الانحدار الاشتقاقيفي الممارسة العملية، نقوم أولاً باستيراد بعض المكتبات.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
from itertools import product
```

للبدء، نحدد دالة موضوعية بسيطة $f(x) = x^2 - 2x - 3$ أن x هي أرقام حقيقية. نظرًا لأن الانحدار الاشتقاقي يستخدم الانحدار، فإننا نحدد أيضًا الانحدار f ، وهو فقط المشتق الأول من f ، يعني $\nabla f(x) = 2x - 2$:

```
def func(x):
    return x**2 - 2*x - 3

def fprime(x):
    return 2*x - 2
```

بعد ذلك، نحدد دوال بايثون لرسم دالة الهدف ومسار التعلم أثناء عملية التحسين:

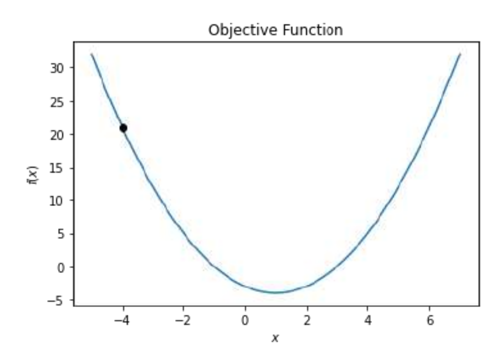
```
def plotFunc(x0):
    x = np.linspace(-5, 7, 100)
    plt.plot(x, func(x))
    plt.plot(x0, func(x0), 'ko')
    plt.xlabel('$x$')
    plt.ylabel('$f(x)$')
    plt.title('Objective Function')

def plotPath(xs, ys, x0):
    plotFunc(x0)
    plt.plot(xs, ys, linestyle='--', marker='o', color='#F12F79')
    plt.plot(xs[-1], ys[-1], 'ko')
```

من الرسم البياني أدناه، يمكننا أن نرى بسهولة أن f لها قيمة دنيا هي $x = 1$. لنفترض أننا بدأنا بـ $x = -4$ (المشار إليها بنقطة سوداء أدناه)، نريد أن نرى ما إذا كان الانحدار الاشتقاقي يمكنه تحديد الحد الأدنى المحلي $x = 1$.



```
x0 = -4
plotFunc(x0)
```



نحدد خوارزمية الانحدار الاشتقاقي البسيط على النحو التالي: لكل نقطة x_k في بداية الخطوة k، نحافظ على طول الخطوة α_k ثابتًا ونضبط اتجاه p_k سلبيًا على قيمة الانحدار. نقوم بتنفيذ هذه الخطوات باستخدام الصيغة التالية:

$$x_{k+1} = x_k + \alpha_k p_k$$

حيث يكون الانحدار أعلى من قيمة تحمل معينة (في حالتنا 1×10^{-5}) ويكون عدد الخطوات قيمة معينة (في حالتنا 1000).

```
def GradientDescentSimple(func, fprime, x0, alpha, tol=1e-5, max_iter=1000):
    # initialize x, f(x), and -f'(x)
    xk = x0
    fk = func(xk)
    pk = -fprime(xk)
    # initialize number of steps, save x and f(x)
    num_iter = 0
    curve_x = [xk]
    curve_y = [fk]
    # take steps
    while abs(pk) > tol and num_iter < max_iter:
        # calculate new x, f(x), and -f'(x)
        xk = xk + alpha * pk
        fk = func(xk)
        pk = -fprime(xk)
        # increase number of steps by 1, save new x and f(x)
        num_iter += 1
        curve_x.append(xk)
        curve_y.append(fk)
    # print results
    if num_iter == max_iter:
        print('Gradient descent does not converge.')
    else:
        print('Solution found:\n  y = {:.4f}\n  x = {:.4f}'.format(fk, xk))
```

```
    return curve_x, curve_y
```

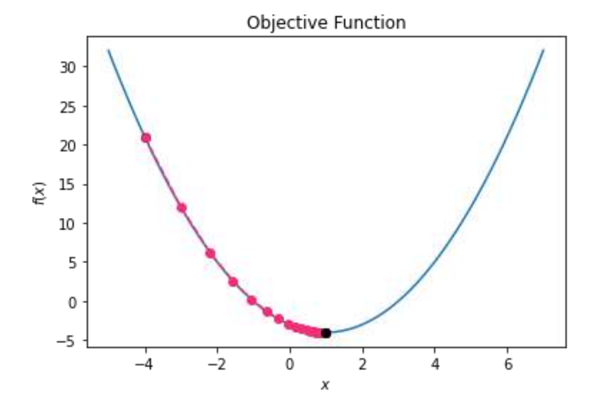
نبدأ من $x = -4$ ونشغل خوارزمية الانحدار الاشتقاقي على f بسيناريوهات مختلفة:

$\alpha_k = 0.1$
$\alpha_k = 0.9$
$\alpha_k = 1 \times 10^{-4}$
$\alpha_k = 1.01$

السيناريو الاول: $\alpha_k = 0.1$

```
xs, ys = GradientDescentSimple(func, fprime, x0, alpha=0.1)
plotPath(xs, ys, x0)
```

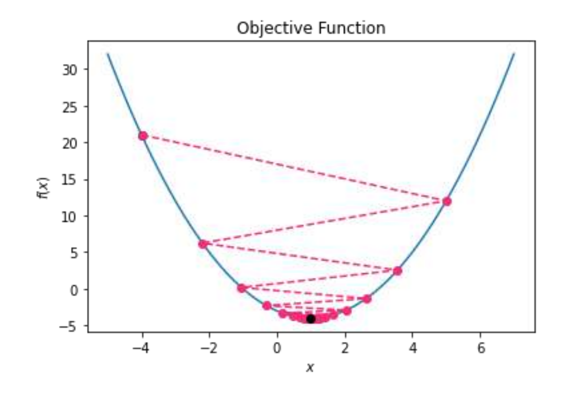
```
Solution found:
  y = -4.0000
  x = 1.0000
```



السيناريو الثاني: $\alpha_k = 0.9$

```
xs, ys = GradientDescentSimple(func, fprime, x0, alpha=0.9)
plotPath(xs, ys, x0)
```

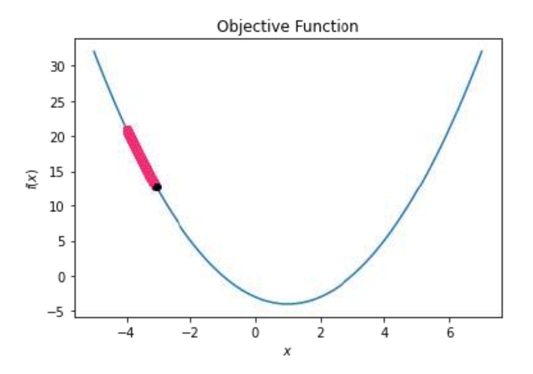
```
Solution found:
  y = -4.0000
  x = 1.0000
```



السيناريو الثالث: $\alpha_k = 1 \times 10^{-4}$

```
xs, ys = GradientDescentSimple(func, fprime, x0, alpha=1e-4)
plotPath(xs, ys, x0)
```

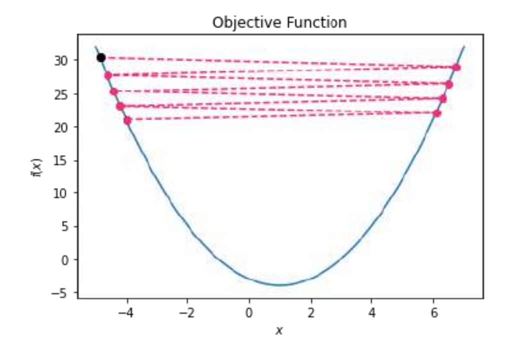
```
Gradient descent does not converge.
```



السيناريو الرابع: $\alpha_k = 0.1$

```
xs, ys = GradientDescentSimple(func, fprime, x0, alpha=1.01, max_iter=8)
plotPath(xs, ys, x0)
```

```
Gradient descent does not converge.
```



يمكن تلخيص ما حصلنا عليه من الرسوم التوضيحية أعلاهفي السيناريوهات المختلفة على النحو التالي:

السيناريو الأول متقارب بسهولة. حتى إذا كان طول الخطوة ثابتًا، فإن الاتجاه ينخفض إلى الصفر، مما يؤدي إلى التقارب.

يتقارب السيناريو الثاني أيضًا على الرغم من تذبذب مسار التعلم بسبب طول الخطوة الكبيرة حول الحل.

السيناريو الثالث يتحرك نحو الحل. ومع ذلك، فإن طول الخطوة صغير جداً بحيث يتم تكبير عدد التكرارات ولا يمكنه العثور على الإجابة.في هذه الحالة، تؤدي زيادة الحد الأقصى إلى حل المشكلة.

السيناريو الرابع مختلف بسبب طول الخطوة الكبيرة. هنا، قمنا بتعيين max_iter = 8 لتحسين الرسم التوضيحي.

شيء واحد يمكن فهمه هو أن الحل x = 1 يمكن الحصول عليه بانحدار اشتقاقي بطول الخطوة المناسب.

قد تتساءل عن سبب عدم استخدامنا لحل تحليلي دقيق: خذ المشتق f ، ثم حل x بحيث تكون المشتقة صفرًا. بالنسبة للمثال السابق، نجد أن x التي تقلل من f تحقق $\nabla f(x) = 2x - 2$ ، أي $x = 1$. نعم، هذه طريقة واحدة. ولكن عندما تواجه مشكلة تحسين حيث يصعب أو يستحيل حل المشتقة f ، لم يعد يوصى بهذه التقنية.

لاحظ أن التنفيذ البسيط أعلاه كان فقط من أجل فهم أفضل لكيفية عمل الانحدار الاشتقاقي مع الرسم التوضيحي.في الممارسة العملية، ليست هناك حاجة للتنفيذ، وتتضمن أطر التعلم العميق التنفيذ الفعال لهذه الخوارزميات.

الانحدار الاشتقاقي المعتمد على الزخم (Momentum)

تأخذ طريقة الانحدار الاشتقاقي القياسي باستخدام معدل التعلم α خطوة صغيرةفي الاتجاه المعاكس للانحدار. هذا المعامل له قيمة ثابتة طوال عملية التعلم. معادلته الرياضية على النحو التالي:

$$\theta_{t+1} = \theta_t - \alpha . \nabla_t(\theta_t)$$

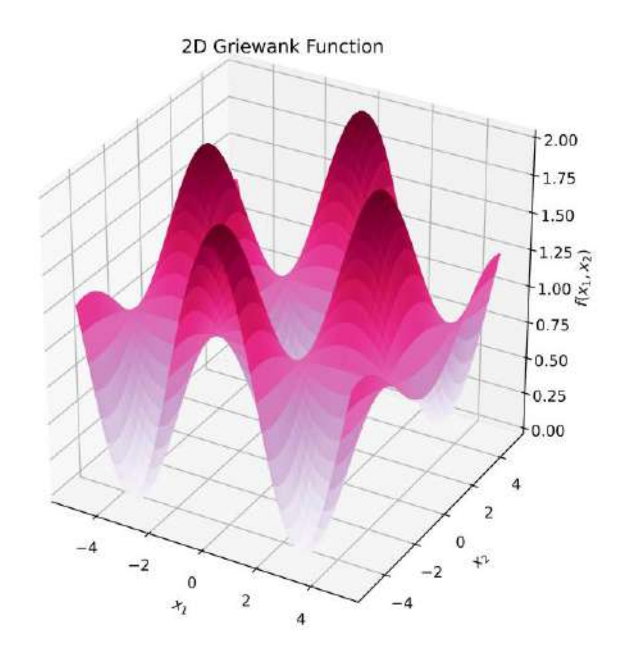
حيث θ_t هي معاملات / أوزان النموذجفي الفترة t ، $\nabla_t(\theta_t)$ هي الانحدار لكل وزن θ_tفي الفترة t و α هي معدل التعلم. يتحرك الانحدار الاشتقاقي القياسي بشكل مستقل عن الخطوات السابقة بحجم خطوة ثابت إلى أسفل. إذا كانت الدالة المستهدفة مثل وادي طويل مع تحسينات محلية وجدران منحدرة على كلا الجانبين (انظر الشكل 3ـ3) ، فسيكون تحديث الأوزان بطيئًا جداً وسيؤدي إلى عدد كبير من الخطوات. لحل هذه المشكلة، يُضاف **الزخم (Momentum)**إلى

خوارزمية الانحدار الاشتقاقي. الفكرة الرئيسية للزخم هي إضافة ذاكرة قصيرة المدى إلى الانحدار الاشتقاقي. بمعنى آخر، بدلاً من استخدام الانحدار للمرحلة الحالية لتوجيه البحث، يقوم الزخم أيضًا بتجميع تدرجات الخطوات السابقة لتحديد الاتجاه. يمكن تنفيذ هذه الآلية على النحو التالي:

$$\theta_{t+1} = \theta_t - v_t$$

$$v_t = \gamma . v_{t-1} - \alpha . \nabla_t(\theta_t)$$

في هذه المعادلات، γ هو تعبير الزخم الذي يحدد تأثير الانحدارات السابقة على التحديث الحالي. الفكرة العامة هي وضع هذه العبارة بالقرب من 1 قدر الإمكان واختيار قيمة عالية قدر الإمكان لمعدل التعلم، مع الحفاظ على تقارب مستقر.



شكل 3ـ3. توضيح دالة Griewank

من خلال إضافة المحفوظات إلى معادلة تحديث المعامل بناءً على الانحدار الذى تمت مواجهته فى التحديثات السابقة، يغير الزخم الانحدار الاشتقاق (يضيف ذاكرة إليه بطريقة ما) مما يؤدى إلى تقارب أسرع.

الانحدار الاشتقاقي المسرع نيستوروف (NAG)

يبدو أن تشبيه الكرة التي تتدحرج إلى أسفل الوادي وتتسارع من خلال الزخم هو ميزة جيدة للتنفيذ. ومع ذلك، لا تتوقف الكرة عندما تكونفي الوادي، فهي تدور أكثر فأكثر حتى تنتهي

السرعة. من ناحية أخرى، إذا كان هناك صعود على الجانب الآخر من الوادي، فسيؤدي ذلك إلى ارتداد الكرةفي النهاية للخلف، ولكن ستكون هناك عدة تقلبات قبل أن تتوقف الكرة عند أدنى نقطةفي الوادي. جاء نيستوروف (1983) بفكرة منع الكرة من الارتفاع عالياً. لحساب حجم الخطوةفي الفترة الحالية، تقوم بحساب التدرج اللونيفي الموقع التقريبي حيث تنتهي الكرة باستخدام الزخم القياسي. بعد ذلك، يتم استخدام هذا الانحدار لإنشاء تصحيح، وهو مناسب بشكل خاصفي الحالات التي يوجد فيها منحدر صعوديفي موقع تقريبي. هذا يمنع حجم الدرجات الكبيرة جداً المؤدية إلى الجانب الآخر من الوادي. يطبق نيستوروف هذه الطريقة على النحو التالي:

$$\theta_{t+1} = \theta_t - v_t$$

$$v_t = \gamma . v_{t-1} - \alpha . \nabla_t(\theta_t - \gamma . v_{t-1})$$

تسمى هذه الطريقة بـ **Nesterov Accelerated Gradient Descent** أو **NAG** للاختصار. عيب هذه الطريقة هو حساب $\nabla_t(\theta_t - \gamma . v_{t-1})$ ، والتي يمكن أن تستغرق وقتًا حسابيًا لبعض الشبكات. يقترح Sutskever وآخرون (2013) تعديل حساب v_t على النحو التالي:

$$v_t = \gamma . m_t + \alpha . \nabla_t(\theta_t)$$

$$m_t = \gamma . m_{t-1} + \alpha . \nabla_t(\theta_t)$$

خوارزمية الانحدار الاشتقاقي في keras

عند استخدام Keras، لتحديد المُحسِّن، يمكن تخصيص مُحسِّن الانحدار الاشتقاقي العشوائي (SGD) عن طريق أخذ عينات مباشرة من فئة SGD واستخدامها عند تجميع النموذج:

```
from tensorflow.keras.optimizers import SGD
...
sgd = SGD(learning_rate=0.0001, momentum=0.8, nesterov=True)
model.compile(optimizer=sgd, loss = ..., metrics= ...)
```

تقبل فئة SGD باراميتر (learning_rate) (معدل التعلم مع الإعداد الافتراضي إلى 0.01)، والزخم(momentum)، ونستروف nesterov بقيمة منطقية ومعامل انحلال (decay) اختيارية.

خوارزميات تحسين معدل التعلم التكيفي

تستخدم جميع إصدارات الانحدار الاشتقاقي التي تم تقديمها حتى الآن نفس معدل التعلم لحساب تحديث كل معامل على حدة. لكن النقطة المهمة هي أن معلومات الانحدارفي تحديث واحد يمكن أن تكون أكثر أهمية لمعامل واحد من أخرى. لذلك،في مثل هذه الحالة، يجب أن يتخذ المعامل ذات التحديثات الأكثر أهمية خطوة أكبرفي اتجاه الانحدار من المعامل ذات

التحديثات الأقل أهمية. بمعنى آخر، لتسريع عملية التعلم، يجب أن يكون لكل معامل يجب تحديثه معدل التعلم الخاص بهفي كل دورة تدريبية.

ومن ثم، تم اقتراح خوارزميات مختلفة لحل هذه المشكلة من أجل تكييف معدل التعلمفي مراحل مختلفة من الخوارزمية لإنشاء تقارب شبكة أسرع. فيما يلي سنراجع بعض هذه الخوارزميات.

Adagrad

يمكن أن يكون العثور على معدل التعلم الأمثل لخوارزمية التعلم العميق مشكلة معقدة. إذا تم تعيين معدل التعلم مرتفعًا جداً، فقد يتقلب المعامل على نطاق واسع للوصول إلى مستوى خطأ مقبول. من ناحية أخرى، فإن تحديد معدل تعليم منخفض للغاية سيؤدي إلى تقدم بطيء للغاية. تم تقديم Adagrad لهذا الغرض، حيث يكون الاختيار اليدوي لمعدلات التعلم المختلفة لكل بُعد من أبعاد المشكلة غير عملي نظرًا لحجم الأبعاد. يقيس Adagrad بشكل نسبي معامل معدل التعلم لكل بُعد للتأكد من أن عملية التعلم ليست بطيئة جداً ولا متقلبة للغاية وغير دقيقة. للقيام بذلك، تجمع خوارزمية AdaGrad ديناميكيًا بين المعرفة الهندسية للبيانات التي لوحظتفي التكرارات السابقة. ثم يطبق هذه المعرفة لضبط معدلات التعلم المنخفضة للحصول على ميزات أكثر شيوعًا ومعدلات تعلم أعلى لميزات نادرة نسبيًا. نتيجة لذلك، يتم تمييز الميزات النادرة وتمكين المتعلم من تحديد الميزات النادرة والتنبؤية العالية.

في هذه الطريقة ، يتم حساب معدل تعلم منفصل لكل معامل نموذج بناءً على التاريخ الكامل لمعامل الانحدار التربيعي للمعامل. يشبه الانحدار التربيعي أهمية الانحدار. يتم حسابها لكل معامل، ويتم حساب معدل التعلم الحالي بقسمة الانحدار الحالي على مربع الانحدار بالإضافة إلى قيمة صغيرة لـ ε (لتجنب القسمة على الصفر). هذا يعني أنه كلما زادت التدرجات التي تم الحصول عليها مسبقًا ، قلت أهمية التدرج الحالي، وبالتالي قل معدل التعلم وبالتالي حجم الخطوة في الدورة التدريبية الحالية. المعادلة الرياضية لهذه الطريقة هي كما يلي:

$$\theta_{t+1} = \theta_t - \alpha . \frac{\nabla_t(\theta_t)}{\sqrt{G_t} + \varepsilon}$$

$$G_t = G_{t-1} + \nabla_t(\theta_t)^2$$

خوارزمية Adagrad في keras

يمكن استخدام مُحسِّن Adagradفي Keras باستخدام مقتطف الكود التالي:

```
from tensorflow.keras.optimizers import Adagrad

...
```

```
adagrad = Adagrad(learning_rate=0.0001, epsilon=1e-6)
model.compile(optimizer=adagrad, loss = ..., metrics= ...)
```

AdaDelta

الفكرة وراء خوارزمية Adagrad جيدة، لكن الخوارزمية لها بعض العيوب. بعد فترة، يصبح تراكم التدرجات المربعة كبيرًا جداً بحيث يكون معدل التعلم المستخدمفي التحديثات منخفضًا جداً وبالتالي لا يمكن إحراز أي تقدم تقريبًا. تحاول خوارزمية AdaDelta معالجة هذه المشكلة عن طريق تقييد نافذة الانحدار الماضي المكدسة إلى حجم ثابت بدلاً من أخذ التاريخ بالكامل. للقيام بذلك، بدلاً من جمع كل الانحدارات المربعة من بداية التعليمات، افترض أننا قمنا بتخزين المجموع المتناقص لهذه التدرجات. يمكننا اعتبار هذا كقائمة مستمرة لأحدث التدرجات لكل وزن.في كل مرة نقوم فيها بتحديث الأوزان، نضع التدرج الجديدفي أسفل القائمة ونحذف الأقدم من البداية. للعثور على القيمة التي نستخدمها لتقسيم الانحدار الجديد، نضيف جميع القيم الموجودةفي القائمة، لكن أولاً نضربها جميعًافي رقم بناءً على موضعهافي القائمة. يتم ضرب القيم الأحدث بمقدار كبير، بينما يتم ضرب القيم الأقدم بمقدار صغير جداً. وبالتالي، يتم تحديد إجمالي التنفيذ لدينا بقوة من خلال التدرجات الحديثة، على الرغم من تأثرها بدرجة أقل بالتدرجات القديمة. باختصار، من أجل التنفيذ الفعال، بدلاً من تخزين التدرجات السابقة المقابلة، يتم استخدام المتوسط المتناقص لجميع التدرجات المربعة السابقة.

تغير هذه الخوارزمية بشكل تكيفي معدل تحديث الأوزان في كل خطوة باستخدام الوزن الإجمالي لكل خطوة. نظرًا لأن Adadelta يضبط مقدار تدريب الوزن بشكل فردي ، فإن أي وزن يبقى على منحدر حاد لفترة من الوقت سوف يتباطأ ليبقى بعيداً عن الشكل ، ولكن عندما يكون هذا الوزن في جانب أكثر سلاسة ، يمكن أن يأخذ خطوات أكبر.

بصرف النظر عن هذا ، يقدم مؤلفو المقال طريقة للتخلص من الحاجة إلى معدلات التعلم من الخوارزمية. يشيرون إلى أن وحدات معدل التعلم ومتوسط الانحطاط لجميع التدرجات التربيعية السابقة غير متطابقة. إنهم يحلون هذه المشكلة عن طريق استبدال معدل التعلم بمتوسط تنازلي آخر من المعامل المربع بتحديث $\Delta\theta_t^2$. هذا مشابه لأهمية تغييرات المعاملات السابقة. بقسمة الجذر الثاني على الجذر التربيعي لأهمية التدرجات السابقة ، نحصل على قيمة أكثر أو أقل نسبة أهمية التدرجات السابقة المستخدمة لتحديث المعاملات ، أو بعبارة أخرى تأثير التدرجات السابقة على القيمة الحالية للمعامل. رمزها الرياضي على النحو التالي:

$$\theta_{t+1} = \theta_t - \nabla_t(\theta_t).\frac{\sqrt{E(\Delta\theta_t^2)_t} + \varepsilon}{\sqrt{E(\nabla^2)_t} + \varepsilon}$$

$$E(\Delta\theta_t^2)_t = \gamma.E(\Delta\theta^2)_{t-1} + (1-\gamma)\,\Delta\theta_t^2$$

$$E(\nabla^2)_t = \gamma.E(\nabla^2)_{t-1} + (1-\gamma)\nabla_t(\theta_t)^2$$

خوارزمية Adadelta في keras

يمكن استخدام مُحسِّن Adadelta في Keras باستخدام مقتطف الكود التالي:

```
from tensorflow.keras.optimizers import Adadelta
...
adadelta= Adadelta(learning_rate=0.0001, epsilon=1e-6)
model.compile(optimizer= adadelta, loss = ..., metrics= ...)
```

RMSprop

تسمى الخوارزمية التي تشبه إلى حد بعيد Adadelta، ولكنها تستخدم رياضيات مختلفة قليلاً، RMSprop. الاسم مشتق من حقيقة أنه يستخدم عملية جذر متوسط التربيع (**root mean-squared**)، وغالبًا ما يتم اختصارها كـ RMS، لتحديد المطابقة التي تمت إضافتها (أو نشرها) إلى التدرجات. رمزها الرياضي على النحو التالي:

$$\theta_{t+1} = \theta_t - \alpha . \frac{\nabla_t(\theta_t)}{\sqrt{E(\nabla^2)_t} + \varepsilon}$$

$$E(\nabla^2)_t = \gamma . E(\nabla^2)_{t-1} + (1-\gamma)\nabla_t(\theta_t)^2$$

خوارزمية RMSprop في keras

باستخدام مقتطف الكود التالي، يمكنك استخدام مُحسِّن RMSprop في Keras:

```
from tensorflow.keras.optimizers import RMSprop
...
rms_prop = RMSprop(learning_rate=0.0001, epsilon=1e-6)
model.compile(optimizer= rms_prop, loss = ..., metrics= ...)
```

Adam

تشترك الخوارزميات السابقة في فكرة تخزين قائمة من التدرجات المربعة من أي وزن. بعد ذلك، من خلال إضافة القيم في هذه القائمة، قد يقومون بإنشاء عامل مقياس بعد قياسها. يتم تقسيم التدرج في كل خطوة من خطوات التحديث على هذا المجموع. يعطي Adagrad نفس الوزن لجميع العناصر الموجودة في القائمة عند إنشاء عامل التحجيم، بينما يعتبر Adadelta و RMSprop أن العناصر الأقدم غير مهمة وبالتالي يكون لها حصة أصغر من الإجمالي.

يعد تربيع التدرج قبل وضعه في القائمة مفيداً رياضيًا، ولكن عندما نربّع رقمًا، تكون النتيجة إيجابية دائمًا. هذا يعني أننا نفقد الاتجاه الإيجابي أو السلبي لذلك التدرج في قائمتنا، وهي معلومات مفيدة. لذلك، لتجنب فقدان هذه المعلومات، يمكننا الاحتفاظ بقائمة ثانية من التدرجات دون تربيعها. يمكننا بعد ذلك استخدام كلتا القائمتين لاشتقاق معامل القياس. هذا نهج حسابي يسمى **تقدير اللحظة التكيفي (adaptive moment estimation)**، أو Adam.

خوارزمية Adam في keras

باستخدام مقتطف الكود التالي، يمكنك استخدام مُحسِّن Adam في Keras:

```
from tensorflow.keras.optimizers import Adam
...
adam= Adam(learning_rate=0.0001, beta_1=0.9, beta_2=0.999, epsilon=1e-6)
model.compile(optimizer= adam, loss = ..., metrics= ...)
```

دالة الخطأ Loss Function

في مرحلة تدريب الشبكة العصبية، يتم استخدام النتيجة (**score**) للإشارة إلى الحالة الحالية. بناءً على هذه النتيجة، يتم البحث عن معاملات الوزن المثلى. بمعنى آخر، تبحث الشبكة العصبية عن المعاملات المثلى باستخدام النقاط كدليل. يتم حساب هذه الدرجة من خلال دالة الخطأ (دالة الخسارة)، بناءً على قياس مقدار الخطأ بين القيم المتوقعة والفعلية. توضح الصيغة البسيطة التالية دالة الخطأ كمعادلة:

$$Loss = y - \hat{y}$$

حيث آن y و $\hat{y}$ هي القيم الفعلية والقيم المتوقعة ، على التوالي.

التعريف 2.3 دالة الخطأ

إنها كمية يتم تقييمها بشكل دوري أثناء التدريب وتعبر عن معدل تقدم التعلم.

يتم تدريب الشبكة العصبية من خلال عملية التحسين وتستخدم دالة الخطأ لحساب الخطأ بين القيمة المتوقعة للنموذج والمخرجات المتوقعة (المخرجات الفعلية). لأغراض التدريب المختلفة، قد تقلل عملية التحسين من دالة الخطأ أو تزيدها إلى أقصى حد، مما يعني أنه يجب عليها تقييم حل مناسب، مثل مجموعة من المعاملات، لتحقيق أدنى درجة أو أعلى درجة، على التوالي.

باستخدام دالة الخطأ ، يمكننا تقييم كيفية نمذجة الخوارزمية على مراجعة البيانات. يعتمد اختيار دالة الخطأ على نوع المشكلة وستختلف دالة الخطأ هذه باختلاف مشاكل التصنيف والانحدار. في مشكلة التصنيف ، نعتزم توقع توزيع احتمالي لمجموعة الفئات. الآن ، في مشاكل الانحدار ، سنجد قيمة معينة.

دوال الخطأ للانحدار

كما ذكرنا سابقًا، تتعامل نماذج الانحدار مع التنبؤ بقيمة مستمرة، مثل سعر السيارة والتنبؤ بالقرض وما إلى ذلك.في هذا القسم، يتم سرد دوال الخطأ للانحدار الأكثر استخدامًا.

Mean Square Error

هي إحدى دوال الخطأ الأكثر شهرة في الانحدار ، وتحسب متوسط الفرق التربيعي بين القيم الفعلية والمتوقعة بالمعادلة التالية:

$$MSE(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

حيث n هو عدد العينات التعليمية.

باستخدام مقتطف الكود التالي، يمكن استخدام دالة MSE في Keras:

```
from tensorflow import keras
...
loss_fn = keras.losses.MeanSquaredError()
model.compile(loss=loss_fn, optimizer=..., metrics= ...)
```

Mean Absolute Error

تحسب هذه الدالة متوسط فرق القيمة المطلقة بين القيم الفعلية والمتوقعة بالمعادلة التالية:

$$MAE(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

باستخدام مقتطف الكود التالي، يمكن استخدام دالة MAE في Keras:

```
from tensorflow import keras
...
loss_fn = keras.losses.MeanAbsoluteError()
model.compile(loss=loss_fn, optimizer=..., metrics= ...)
```

دوال الخطأ للتصنيف

في هذا القسم، يتم وصف دوال الخطأ المتعلقة بالتصنيف.

Cross Entropy

تحسب هذه الدالة المسافة بين توزيعين احتماليين ويتم تعريفها على النحو التالي:

$$Cross\ Entropy(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y})$$

باستخدام مقتطف الكود التالي، يمكن استخدام دالة الخطأ Cross Entropy في Keras:

```
from tensorflow import keras

...
```

```
loss_fn = keras.losses.CategoricalCrossentropy()
model.compile(loss=loss_fn, optimizer=..., metrics= ...)
```

Binary Cross Entropy

يتم استخدام Binary Cross Entropy للمصنفات الثنائية، والتي يتم تعريفها على النحو التالي:

$$Binary\ Cross\ Entropy(y, \hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}(y_i \log(\hat{y}) + (1 - y_i)(1 - \log(\hat{y})))$$

باستخدام مقتطف الكود التالي، يمكن استخدام دالة الخطأ Binary Cross Entropyفي Keras:

```
from tensorflow import keras
...
loss_fn = keras.losses.binary_crossentropy
model.compile(loss=loss_fn, optimizer=..., metrics= ...)
```

الانتشار الخلفي Backpropagation

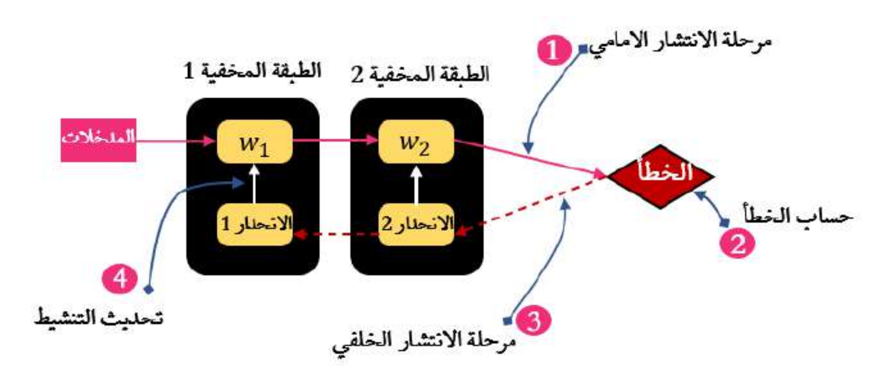
التعلمفي شبكة عصبية متعددة الطبقات يشبه بشكل عام تعلم بيرسيبترون، مع الأوزان المتطابقة. ومع ذلك، فإن الطريقة التي يجب أن يتغير بها كل وزن تكون أكثر صعوبة بقليل.في حالة بيرسيبترون، فإن الضرب الداخلي فقط بين بيانات الإدخال والأوزان هو المطلوب للحصول على ناتج العقدة. نظرًا لوجود طبقات متعددةفيMLP ، يتم تحديد ناتج العقدةفي الطبقة الأخيرة من خلال أخذ الضرب الداخلي للأوزان ومخرجات العقدفي الطبقة السابقة. يتم حساب كل من هذه القيم الحديثة بنفس الطريقة. هذا يعني أن الخطأفي إخراج الطبقة النهائية يمكن أن يكون بسبب أوزان الطبقة الأخيرة بالإضافة إلى أوزان الطبقة (الطبقات) السابقة. لذا فإن السؤال هو وزن أي طبقة سبب هذا الخطأ. يشار إلى هذه المشكلة أحيانًا باسم **تخصيص الائتمان** (**credit assignment**). الطريقة التي يمكن أن تحل هذه المشكلة تسمى **الانتشار الخلفي**.

ربما تكون خوارزمية الانتشار الخلفي هي اللبنة الأساسيةفي الشبكة العصبية. يعد الانتشار الخلفيفي الأساس طريقة ذكية لحساب التدرجات بشكل فعالفي الشبكات العصبية متعددة الطبقات. تستخدم هذه الخوارزمية قاعدة حساب التفاضل والتكامل وتحسب تدرج الخطأفي مسارات مختلفة من عقدة واحدة إلى المخرجات وتتكون من مرحلتين رئيسيتين تسمى مرحلة الانتشار الامامي ومرحلة الانتشار الخلفي.في هذه الخوارزمية، بعد كل تمريرة للأمامفي الشبكة، تقوم مرحلة الانتشار الخلفي بإجراء تمريرة عكسية وفي نفس الوقت تعدل معاملات النموذج (الأوزان والتحيزات).

تسمح خوارزمية الانتشار الخلفي بحساب الانحدار لطبقة واحدة في كل مرة بكفاءة، واستخدام القيم المعاد حسابها ، وعكس الطبقة الأخيرة من خلال قانون السلسلة.

باختصار، على مستوى عالٍ جدًا، بناءً على ما قيل حتى الآن، تكرر الشبكة العصبية هذه الخطوات وتنفذها عدة مرات أثناء التدريب. (الشكل 3ـ4):

- **مرحلة الانتشار الامامي لتوليد المخرجات بناءً على المعاملات الحالية وبيانات الإدخال.**
- **مرحلة حساب دالة الخطأ للفرق بين المخرجات الحالية والهدف.**
- **مرحلة الانتشار الخلفي لحساب الانحدار للخطأ بالنسبة للمعاملات.**
- **مرحلة تحسين التدرجات لتحديث المعاملات لتقليل الخسائر للتكرار التالي.**



الشكل 3ـ4. عملية تدريب الشبكة العصبية

تهيئة الأوزان للمعاملات

عنصر مهم جدًافي تصميم الشبكات العصبية هو تهيئة الأوزان.في الشبكات العصبية، يجب اختيار القياس الكمي الأولي للأوزان بعناية فائقة. على سبيل المثال، إذا كان للعديد من الخلايا العصبيةفي طبقة مخفية نفس الأوزان، فسوف تتلقى نفس التدرجات. نتيجة لذلك، يتم تحسين جميع الخلايا العصبية بنفس الطريقة عند تصحيح التدرج. بمعنى آخر، يتم حساب نفس النتائج لكل منهم، مما يؤدي إلى إهدار سعة النموذج. وبالتالي، فإن الشبكة المكافئة لسلسلة من الطبقات هي أحادية العصبونات.

إذا عرفنا التكوين النهائي (أو حتى تقريبًا)، فيمكننا تعديلها للوصول بسهولة إلى النقطة المثلى في عدد قليل من التكرارات. لكن، للأسف، لا نعرف أين هو نقطة المحلي الامثل. ومن ثم، تم تطوير واختبار استراتيجيات تجريبية بهدف تقليل وقت التدريب. من الناحية المثالية، يجب أن تظل تباينات التنشيط ثابتةفي جميع أنحاء الشبكة تقريبًا بالإضافة إلى تباينات الوزن بعد كل خطوة الانتشار الخلفي. هذان الشرطان ضروريان لتحسين عملية التقارب ولمنع مشاكل تلاشي التدرج والانفجار.

عادة، يتم تهيئة أوزان الشبكة العصبية باستخدام توزيع غاوسي بمتوسط صفر وانحراف معياري صغير. ومع ذلك، فإن المشكلة تكمن في أن توزيع ناتج خلية عصبية تمت تهيئتها عشوائيًا له تباين يزيد مع عدد المدخلات. لتنعيم تباين الإخراج لكل خلية عصبية إلى 1، يكفي استخدام التوزيع الطبيعي القياسي وقياس الوزن بناءً على الجذر التربيعي لسعة الإدخال n_{in}، وهو عدد المدخلات:

$$w_0 \sim \frac{\mathcal{N}(0,1)}{\sqrt{n_{in}}}$$

تسمى هذه الطريقة **مقياس التباين** (**variance scaling**)ويمكن تطبيقها بالإضافة إلى عدد وحدات الإدخال (Fan-In) ، بناءً على عدد وحدات الإخراج (Fan-Out) أو متوسطها. هذه الفكرة بديهية للغاية ، إذا كان عدد وصلات الإدخال أو الإخراج كبيرًا ، فيجب أن تكون الأوزان أصغر لتجنب المخرجات الكبيرة.

يمكن استخدام variance scaling باستخدام مقتطف التعليمات البرمجية التالي في Keras:

```
from keras import initializers
initializer = initializers.VarianceScaling(scale=1.0, mode='fan_in',
distribution='normal')
```

حلل Gloret و Benjio الانحدارات بعد الانتشار وأوصيا بالتهيئة (المعروفة باسم تهيئة Xavier):

$$w_0 \sim \sqrt{\frac{2}{n_{in} + n_{out}}}\mathcal{N}(0,1)$$

حيث يصف n_{out} عدد وحدات الإخراج. هذا نوع أقوى من الطريقة السابقة ، لأنه يأخذ في الاعتبار كل من اتصالات الإدخال والإخراج (والتي بدورها هي اتصالات الإدخال). الهدف هو محاولة تلبية المطلبين المقدمين سابقًا. لقد ثبت أن تهيئة Xavier فعالة للغاية في العديد من البنى العميقة وغالبًا ما تكون الخيار الافتراضي.

يمكن استخدام Xavier في Keras باستخدام مقتطف الشفرة البرمجية التالي:

```
from keras import initializers
initializer = initializers.GlorotNormal()
```

الغرض من تهيئة الأوزان هو منع مخرجات دوال التنشيط من الانفجار أو التلاشي على طول المسار عبر شبكة عصبية عميقة. إذا حدث أحد هذين الأمرين، فإن تدرجات الخطأ (الخسائر) ستكون إما كبيرة جدًا أو صغيرة جدًا بحيث لا يمكن أن تتدفق إلى الوراء بشكل مفيد. حتى لو تمكنت من القيام بذلك على الإطلاق، فإن الشبكة تحتاج إلى مزيد من الوقت للتقارب.

كل هذه الطرق مشتركة بينهم ويمكن تبادلها فى كثير من المواقف، كما ذكرنا سابقًا، فإن Xavier's أحد أقوى المضيفين فى جميع الجوانب لديه دالة جيدة، ومع ذلك، يمكن فقط التحقق من صحة مجموعة البيانات الفعلية لتأكيد ذلك.

المعاملات

في أي شبكة عصبية عميقة، هناك نوعان مختلفان من المعاملات: أحدهما هو معامل النموذج والآخر هو المعامل الفائق (**hyper parameter**). معاملات النموذج هي تلك المعاملات التي يتم اكتشافها تلقائيًا بواسطة النموذج من بيانات التدريب.في المقابل، المعاملات الفائقة هي المعاملات يمكن تعديلها لتحقيق أفضل أداء للنموذج. بمعنى آخر، لا يتم تعلم هذه المعاملات أثناء التدريب، ولكن يتم تعيينها من قبل المستخدمفي بداية عملية التعلم. تؤثر المعاملات الفائقة على عملية التعلم بأكملهافي الشبكة العصبية. تتضمن بعض المعاملات الفائقة عدد الطبقات المخفية التي تحدد بنية الشبكة. معدل التعلم هو معامل فائق أخر يساعد على فهم كيفية تدريس الشبكات. يلعب اختيار المعامل الفائق الامثل دورًا مهمًافي عملية تدريب الشبكة بأكملها.

التعريف 3.3 المعامل

تعد المعاملات الفائقة وسيطات نموذجية يتم تعيين قيمها قبل بدء عملية التعلم.

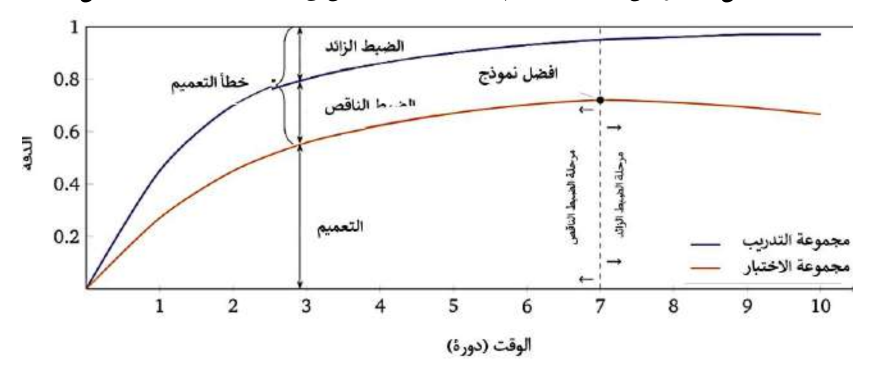
يمكن اعتبار المعاملات الفائقة بمثابة لوحة معلومات بها مفاتيح واتصال يتحكم فى كيفية عمل الخوارزمية، غالبًا ما يؤدى ضبط المعاملات الفائقة المختلفة إلى نماذج ذات أداء مختلف.

التعميم والتنظيم

الغرض الرئيسي من بناء نموذج التعلم العميق هو تعلم وظيفة أو مهمة (task). لتحقيق هذا الهدف، نقوم بتغذية شبكة بيانات التدريب. بعد تدريب شبكة عصبية بالانحدار الاشتقاقي والانتشار الخلفي، نفترض أن هذا الأداء الجيد يظل على بيانات غير مرئية (أي البيانات التي لم يتم تضمينها أثناء التدريب). ومع ذلك، هذا لا يعني بالضرورة أن النموذج قادر على التنبؤ بدقة بإخراج البيانات غير المرئية. لذلك، يتم تقديم مجموعتين إضافيتين من البيانات للتحسين، مجموعة التحقق من الصحة والمجموعة التجريبية. كل مجموعات البيانات الثلاث مستقلة عن بعضها البعض، لذلك لا توجد حالات مشتركة بينها.

تُستخدم مجموعات التحقق من الصحة في الشبكات العصبية بشكل شائع لضبط المعاملات للنموذج مثل بنية الشبكة أو معدلات التعلم. تستخدم مجموعة الاختبار فقط للتقييم النهائي من أجل تقييم أداء الشبكة في البيانات غير المرئية. إذا لم تكن الشبكة العصبية معممة بشكل جيد (القدرة على نقل المعرفة إلى البيانات غير المرئية) ، أي أن فقدان التدريب أقل من فقدان

الاختبار، وهذا ما يسمى الضبط الزائد overfitting. يُطلق على السيناريو العكسي ، عندما تكون خطأ الاختبار أقل بكثير من خطأ التدريب ، الضبط الناقص underfitting. (الشكل 3ـ5).



الشكل 3ـ5. سلوك التعميم في منحنى التعلم وفق معيار الدقة في البيانات التدريبية والاختبارية

الضبط الزائد overfitting هو ظاهرة تؤثر في النهاية على جميع الشبكات العصبية. هذا يرجع إلى حقيقة أنهم يتعلمون فقط من مجموعات البيانات التدريبية: مجموعة فرعية من جميع البيانات الممكنة. يحدد مدى جودة أدائهم في هذه المجموعة الفرعية مقدار المكافأة أو المعاقبة على أوزانهم. بمعنى آخر ، حتى لو كان هدفنا هو المعرفة المعممة ، فإن الشبكات نفسها مصممة لتحقيق دقة عالية في مجموعات بيانات محددة. بهذه الطريقة ، سيبدأون في النهاية في الحفاظ على مجموعة البيانات الخاصة بهم ، بالنظر إلى قدرتهم على القيام بذلك. يؤدي هذا الحفظ إلى زيادة التحميل على الشبكة. تتذكر الشبكة التي بدأت في التجاوز الميزات الفريدة والتفاصيل المحددة للبيانات الموجودة حصريًا في مجموعة بيانات التدريب وتخلط بينها كمفاهيم عامة مشتركة بين جميع مدخلات البيانات المماثلة. لذلك ، ستواجه مثل هذه الشبكة وقتًا أكثر صعوبة في تحليل المدخلات الجديدة وغير المألوفة (مجموعات البيانات التجريبية). وذلك لأن بعض السمات المميزة التي تم تحديدها مسبقًا غير موجودة في البيانات الجديدة. بالإضافة إلى ذلك ، مع زيادة دقة الشبكة أثناء التدريب، تزداد الفجوة بين الخطأ الناتج أثناء التدريب والخطأ الناتج أثناء الاختبار. ومع ذلك ، في بعض الأحيان يكون استخدام نموذج معقد أمرًا لا مفر منه عند محاولة حل المهام المعقدة للغاية. تسمح إضافة المزيد من الطبقات أو الخلايا العصبية بمستوى أعلى من استخراج الميزات ، مما قد يؤدي إلى مستوى أعلى من الدقة إلى نقطة معينة.

عادةً ما يرتبط overfitting و underfitting في الشبكات العصبية العميقة ارتباطًا مباشرًا بقدرة النموذج. ببساطة ، ترتبط سعة النموذج لشبكة عصبية عميقة ارتباطًا مباشرًا بعدد المعاملات داخل الشبكة. تحدد سعة النموذج مدى قدرة الشبكة العميقة على ملاءمة مجموعة واسعة من الوظائف. إذا كانت السعة منخفضة جدًا ، فقد لا تتمكن الشبكة من تكييف مجموعة التدريب (أقل من اللازم) ، بينما قد تؤدي سعة النموذج الكبيرة جدًا إلى الاحتفاظ بعينات التدريب (زيادة العرض). عادة لا يمثل عدم الملائمة مشكلة بالنسبة للشبكات العصبية العميقة.

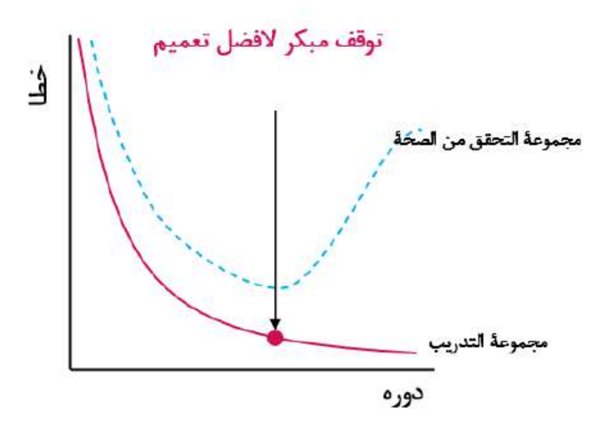
هذا لأنه يمكن حل هذه المشكلة باستخدام بنية شبكة أقوى أو أعمق مع المزيد من المعاملات. ومع ذلك ، من أجل التمكن من استخدام الشبكات العميقة للبيانات الجديدة وغير المرئية ، يجب التحكم في overfitting. تسمى عملية تقليل تأثير overfitting أو منعه **بالتنظيم(regularization)**. التنظيم هو وسيلة للتحكم في overfitting ، أو بالأحرى **لتحسين خطأ التعميم**.

لا ينبغي التغاضي عن ملاءمة البيانات التعليمية. لأن النجاح في التعميم أو حتى الملاءمة الكافية في البيانات التعليمية يعتمد عليها. خلاف ذلك ، قد يميل النموذج إلى التكيف بشكل مفرط مع الميزات المحددة لبيانات التدريب. يعتمد هذا ، من ناحية ، على كمية البيانات المتاحة للتدريب ، حيث قد لا تكون مجموعة التدريب الصغيرة كافية لتحديد الأنماط والهياكل العامة ، ومن ناحية أخرى ، على جودة بيانات التدريب ؛ خاصة عندما يتعلق الأمر بالتعلم من خلال مراقبة صحة علامات الهدف التي تم تعيينها بالفعل من قبل البشر أو حتى الخبراء البشريين. بالإضافة إلى ذلك ، من الضروري التأكد من أن توزيع وخصائص بيانات التدريب متوافقة مع بيانات الاختبار أو أنها متوافقة بشكل عام مع البيانات التي تم التخطيط للنموذج الذي تم تعلمه لاستخدامه في المستقبل.

التوقف المبكر (Early stopping)

في تدريب النموذج الكبير، عادةً ما ينخفض خطأ التدريب والتحقق من الصحة بمرور الوقت، ولكنفي مرحلة ما، يبدأ خطأ التحقق من الصحةفي الزيادة.في هذه المرحلة، يبدأ النموذجفي overfitting ومعرفة الميزات المحددة لمجموعة التدريب. يتم استخدام طريقة التوقف المبكر للتوقف عند هذه المرحلة. تقوم هذه الطريقة بإرجاع النموذج بأقل خطأ تحقق من الصحة. لذلك، يتطلب التدريب مجموعة من عمليات التحقق من الصحة لتقييم أخطاء التحقق بشكل دوري.

في هذه الطريقة، لا يتوقف التدريب بعد الزيادة الأولىفي خطأ التحقق من الصحة. وبدلاً من ذلك، تتلقى الشبكة مزيداً من التدريب حتى تصل إلى عتبة "**عدد الدورات دون تقدم**". بعد ذلك، من خلال تقييم الدورات اللاحقة، تتلقى عملية التحقق من الخطأ لمزيد من التدريب. على سبيل المثال، إذا لم يتحسن خطأ التحقق من الصحة 10 مرات متتالية على أفضل خطأ تحقق من الصحة، يتم إيقاف التدريب ويتم إرجاع النموذج الذي يحتوي على أفضل خطأفي التحقق من الصحة.

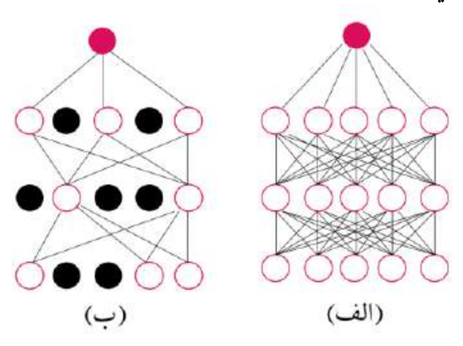


الشكل 3ـ6. **التوقف المبكر**

> يراقب التوقف المبكر أداء النموذج في كل دورة بناءً على مجموعة التحقق ويعتبر نهاية التدريب مشروطة بأداء التحقق.

الحذف العشوائي (Dropout)

الحذف العشوائي هو طريقة تنظيمية للشبكات العصبية. الفكرة الأساسية هي إزالة الخلايا العصبية بشكل عشوائي عند كل تكرار. إذا تم حذف خلية عصبية، فإن جميع التدرجات التابعة تكون صفرية وبالتالي لا يتم عرض الوزن المقابل. إن عملية تنفيذ هذه الطريقة هي أنه في كل فترة تكرار للتدريب، تظل كل خلية عصبية مع احتمال p ، ومع احتمال $(1-p)$ ستتم إزالتها من الشبكة. سيؤدي هذا إلى عدم التعرف على جميع الميزات في كل فترة، وسيتم استخدام خصائص مختلفة للتصنيف في كل مرة يتم فيها إدخال البيانات. يوضح الشكل 3ـ4 شبكة عصبية نموذجية وشبكة عصبية مع حذف عشوائي.



الشكل 3ـ6. **شبكة عصبية (أ) قبل الحذف العشوائي و (ب) بعد الحذف العشوائي**

التفسير الأكثر شيوعًا لتأثير الحذف العشوائي هو أنه يعلم ضمنيًا **مجموعة (ensemble)** من النماذج. لأن كل تكرار ينتج نسخة مختلفة من النموذج ويتم تحديث كل وزن بمجموعة من

الأوزان الأخرى. يعد التعلم الجماعي للعديد من النماذج أسلوبًا شائعًافي التعلم الآلي لتقليل خطأ التعميم بفكرة أن التنبؤ غير الصحيح لنموذج واحد يمكن تعويضه بواسطة نماذج أخرى.

من ناحية أخرى، يمكن تفسير الحذف العشوائي على أنه يتم إجبار الشبكة العصبية على المبالغةفي تمثيل المعرفة المكتسبة من خلال التعلم. نظرًا لأن بعض المعرفة حول فئات أو مدخلات معينة ليست متاحة بالضرورةفي بعض التكرارات، فإن المعرفة المشفرةفي هذه الخلايا العصبية لهذه المدخلات يتم إزالتها الآن. وبالتالي، من الصعب على الشبكة العصبية أن تملأ عينات تدريب محددة، لأن بعض الخلايا العصبية ليست متاحة دائمًا.

يمكن الوصول إلى الحذف التصادفي باستخدام مقتطف الشفرة التاليفي Keras:

```
from keras.layers import Dropout
from keras.models import Sequential

model = Sequential()
...
model.add(Dropout(0.5))
```

التسوية الجماعية (Batch Normalization)

يؤدي تدريب الشبكة إلى تغيير أوزان كل طبقة. يتسبب هذا التغييرفي تغيير توزيع **(distribution)** المدخلات أثناء ترقية الطبقات السابقة، ويسمى هذا التأثير **إزاحة المتغير الداخلي (internal covariate shift)**. تنشأ هذه المشكلة لأن المعاملات تتغير باستمرار أثناء عملية التدريب، والتي بدورها تغير قيم دوال التنشيط. يؤدي تغيير قيم الإدخال من الطبقات الأولية إلى الطبقات التالية إلى تقارب أبطأ أثناء عملية التدريب، لأن بيانات التدريب للطبقات اللاحقة غير مستقرة. بمعنى آخر، الشبكات العميقة هي مزيج من عدة طبقات ذات وظائف مختلفة، وكل طبقة لا تتعلم فقط كيفية تعلم التمثيل العام من بداية التدريب، ولكن عليها أيضًا أن تتقن التغييرات المستمرةفي توزيعات المدخلات وفقًا لما سبق. طبقات. بينما يقوم المُحسِّن بتحديث المعاملات على افتراض أنها لا تتغيرفي الطبقات الأخرى ويقوم بتحديث كل الطبقاتفي نفس الوقت، سيؤدي هذا إلى نتائج غير مرغوب فيها عند الجمع بين دوال مختلفة. من أجل التعامل مع التغييرفي التوزيع أثناء التعلم، تم تقديم **التسوية الجماعية (Batch Normalization)**في هذه الطريقة، يتم إجراء التسوية على بيانات الإدخال أحادية الطبقة بحيث يكون لها متوسط صفر وانحراف معياري بواحد. من خلال وضع التسوية الجماعية بين الطبقات المخفية وإنشاء خاصية تباين مشتركة ، نقوم بتقليل التغييرات الداخلية لطبقات الشبكة.

من خلال تطبيق التسوية الجماعية، يمكن زيادة معدل التعلم، وهذا يؤدى إلى تعلم أسرع. بالإضافة إلى ذلك، تتزايد الدقة مقارنة بنفس الشبكة دون التسوية الجماعية.

يمكن استخدام التسوية الجماعية باستخدام مقتطف الشفرة التاليفي Keras:

```
from keras.layers import BatchNormalization
from keras.models import Sequential

model = Sequential()
...
model.add(BatchNormalization())
```

تنفيذ الشبكة العصبية في keras

في هذا القسم، سوف نتعلم كيفية تنفيذ شبكة عصبية امامية التغذيةفي Keras في المثال الأول، قمنا ببناء شبكة عصبية للتنبؤ بما إذا كانت أسعار المنازل أعلى أو أقل من المتوسط.

مجموعة البيانات التي سنستخدمها مقتبسة من بيانات مسابقة [1] Kaggleللتنبؤ بقيمة منزل Zillow. لقد قللنا عدد ميزات الإدخال وغيرنا العمل للتنبؤ بأن سعر المنزل سيكون أعلى أو أقل من المتوسط. استخدم الرابط الموجودفي الحاشية السفلية[2] لتنزيل مجموعة البيانات المعدلة.

قبل البرمجة لبناء أي نموذج للتعلم الآلي، فإن أول شيء يتعين علينا القيام به هو وضع بياناتنا في تنسيق مناسب للخوارزمية.في هذا المثال، نقوم بما يلي:

- نقرأ أولاً ملفات CSV ونحولها إلى مصفوفات. المصفوفات هي تنسيقات بيانات يمكن للخوارزمية معالجتها.
- نقسم مجموعة البيانات الخاصة بنا إلى خصائص إدخال تسمى x وتسمية (اخراج) تسمى y.
- نقوم بتسوية البيانات.
- نقسم مجموعة البيانات إلى مجموعة تدريب ومجموعة التحقق ومجموعة الاختبار.

هيا بنا نبدأ! أدخل أولاً مكتبة pandas، واكتب الكود التاليفي خلية notebook واضغط على Alt + Enter:

```
import pandas as pd
```

هذا يعني فقط أنني إذا أردت الإشارة إلى الكود الموجودفي حزمة "pandas "، فسأشير إليه على أنه pd. ثم نقرأ ملف CSV الخاص بنا عن طريق تنفيذ الكود التالي:

```
df = pd.read_csv('housepricedata.csv')
```

يعني هذا السطر من التعليمات البرمجية أننا نقرأ ملف `'housepricedata.csv'` ونحفظهفي متغير df. إذاكنت تريد معرفة ما هو موجودفي df ، فما عليك سوى كتابة dfفي خلية notebook واضغط على Alt + Enter:

```
df
```

[1] https://www.kaggle.com/c/zillow-prize-1/data

[2] https://drive.google.com/file/d/1h6LPHNs4F_FnxwfdE_fCIsGeEh30tDBf/view?usp=sharing

يجب أن تبدو مخرجاتك مثل هذا:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAbvGr	TotRmsAbvGrd	Fireplaces	GarageArea	AboveMedianPrice
0	8450	7	5	856	2	1	3	8	0	548	1
1	9600	6	8	1262	2	0	3	6	1	460	1
2	11250	7	5	920	2	1	3	6	1	608	1
3	9550	7	5	756	1	0	3	7	1	642	0
4	14260	8	5	1145	2	1	4	9	1	836	1
...	...	...	...	...	...	...	...	...	...	...	...
1455	7917	6	5	953	2	1	3	7	1	460	1
1456	13175	6	6	1542	2	0	3	7	2	500	1
1457	9042	7	9	1152	2	0	4	9	2	252	1
1458	9717	5	6	1078	1	0	2	5	0	240	0
1459	9937	5	6	1256	1	1	3	6	0	276	0

1460 rows × 11 columns

خصائص الإدخال لدينا موجودةفي الأعمدة العشرة الأولى.في العمود الأخير لدينا الميزة (التسمية) التي نريد توقعها: هل سعر المنزل أعلى من المتوسط أم لا؟ (1 لـ "نعم" و "0" لـ "لا"). الآن بعد أن رأينا شكل البيانات، نريد تحويلها إلى مصفوفات حتى يقوم الجهاز بمعالجتها:

```
dataset = df.values
```

لتحويل إطار البيانات الخاص بنا إلى مصفوفة، نقوم بتخزين قيم df فقط (مع df.values)في متغير dataset . لمعرفة ما يوجد داخل متغير "dataset"، ما عليك سوى كتابة datasetفي خلية notebook وتشغيل الخلية (Alt + Enter):

```
dataset
```

كما ترى، يتم تخزينها جميعًافي مصفوفة الآن:

```
array([[ 8450,     7,     5, ...,     0,   548,     1],
       [ 9600,     6,     8, ...,     1,   460,     1],
       [11250,     7,     5, ...,     1,   608,     1],
       ...,
       [ 9042,     7,     9, ...,     2,   252,     1],
       [ 9717,     5,     6, ...,     0,   240,     0],
       [ 9937,     5,     6, ...,     0,   276,     0]])
```

نقسم الآن مجموعة البيانات الخاصة بنا إلى سمات الإدخال (X) والسمة التي نريد توقعها (Y). للقيام بهذا القسمة، نقوم ببساطة بتعيين الأعمدة العشرة الأولى من المصفوفة إلى متغير يسمى X والعمود الأخير من المصفوفة إلى متغير يسمى Y. كود القيام بالمهمة الأولى كما يلي:

```
X = dataset[:,0:10]
```

قد يبدو هذا غريباً بعض الشيء، لكن اسمحوا لي أن أشرح ما بالداخل. كل شيء قبل الفواصل (،) يشير إلى صفوف المصفوفة وكل شيء بعد الفواصل يشير إلى أعمدة المصفوفة. نظرًا لأننا لا نفصل الصفوف، فإننا نضع ":" قبل الفاصلة. هذا يعني أخذ جميع صفوف مجموعة البيانات ووضعهافي X. نريد استخراج الأعمدة العشرة الأولى، لذلك يعني "0:10" بعد الفاصلة أخذ الأعمدة من 0 إلى 9 ووضعهافي X (لم يتم تضمين العمود 10). تبدأ الاعمدة من الفهرس 0، لذا فإن الأعمدة العشرة الأولى هي أعمدة من 0 إلى 9.

ثم نقوم بتعيين العمود الأخير من مصفوفتنا إلى Y.:

```
Y = dataset[:,10]
```

لقد قسمنا الآن مجموعة البيانات الخاصة بنا إلى خصائص الإدخال (X) والعلامات، أي ما نريد توقعه (Y). الخطوة التاليةفي المعالجة هي التأكد من أن مقياس خصائص الإدخال هو نفسه. حاليًا، الميزات مثل مساحة الأرضية بالآلاف، وتتراوح نقاط الجودة الإجمالية من 1 إلى 10، وعدد المواقد هو 0 أو 1 أو 2. هذا يجعل من الصعب بدء الشبكة العصبيةفي وقت مبكر، مما يسبب بعض المشاكل العملية. تتمثل إحدى طرق قياس البياناتفي استخدام حزمة scikit-Learn. ندخله أولا:

```
from sklearn import preprocessing
```

يقول الكود أعلاه أنني أريد استخدام كود المعالجة المسبقةفي حزمة sklearn. بعد ذلك، نستخدم دالة تسمى مقياس min-max الذي يقيس مجموعة البيانات بحيث تكون جميع خصائص الإدخال بين 0 و1.:

```
min_max_scaler = preprocessing.MinMaxScaler()
X_scale = min_max_scaler.fit_transform(X)
```

لاحظ أننا اخترنا 0 و1 للمساعدةفي تدريب شبكتنا العصبية. يتم الآن تخزين مجموعة البيانات المقاسة لدينافي مصفوفة X_scale. إذاكنت تريد أن ترى كيف يبدو X_scale ، فما عليك سوى تشغيل الخلية التالية:

```
X_scale
```

بعد تنفيذ الكود أعلاه، سترى الإخراج التالي:

```
array([[0.0334198 , 0.66666667, 0.5       , ..., 0.5       , 0.        ,
        0.3864598 ],
       [0.03879502, 0.55555556, 0.875     , ..., 0.33333333, 0.33333333,
        0.32440056],
       [0.04650728, 0.66666667, 0.5       , ..., 0.33333333, 0.33333333,
        0.42877292],
       ...,
       [0.03618687, 0.66666667, 1.        , ..., 0.58333333, 0.66666667,
        0.17771509],
       [0.03934189, 0.44444444, 0.625     , ..., 0.25      , 0.        ,
        0.16925247],
       [0.04037019, 0.44444444, 0.625     , ..., 0.33333333, 0.        ,
        0.19464034]])
```

الآن، وصلنا الى المرحلة الأخيرةفي معالجة البيانات، وهي تقسيم مجموعة البيانات إلى مجموعة التدريب ومجموعة التحقق ومجموعة الاختبار. لهذا سنستخدم الكود scikit-Learn المسمى "train_test_split"، والذي، كما يوحي اسمه، يقسم مجموعة البيانات الخاصة بنا إلى مجموعة تدريب ومجموعة تجريبية. أولاً نقوم بإدخال الكود الذي نحتاجه:

```
from sklearn.model_selection import train_test_split
```

ثم نقسم مجموعة البيانات الخاصة بنا على النحو التالي:

```
X_train, X_val_and_test, Y_train, Y_val_and_test =
train_test_split(X_scale, Y, test_size=0.3)
```

يخبر مقتطف الكود أعلاه scikit-Learn أن حجم `val_and_test` سيكون 30٪ من إجمالي مجموعة البيانات. يخزن الكود البيانات المقسمة في أول أربع متغيرات على يسار علامة التساوي. لسوء الحظ، تساعدنا هذه الدالة فقط في تقسيم مجموعة البيانات الخاصة بنا إلى جزأين. نظرًا لأننا نريد مجموعة تحقق منفصلة ومجموعة اختبار، يمكننا استخدام نفس الدالة لإعادة التوزيع في `val_and_test`:

```
X_val, X_test, Y_val, Y_test = train_test_split(X_val_and_test,
Y_val_and_test, test_size=0.5)
```

الكود أعلاه يقسم حجم `val_and_test` بالتساوي في مجموعة التحقق ومجموعة الاختبار. باختصار، لدينا الآن إجمالي ست متغيرات لمجموعة البيانات التي سنستخدمها:

- X_train
- X_val
- X_test
- Y_train
- Y_val
- Y_test

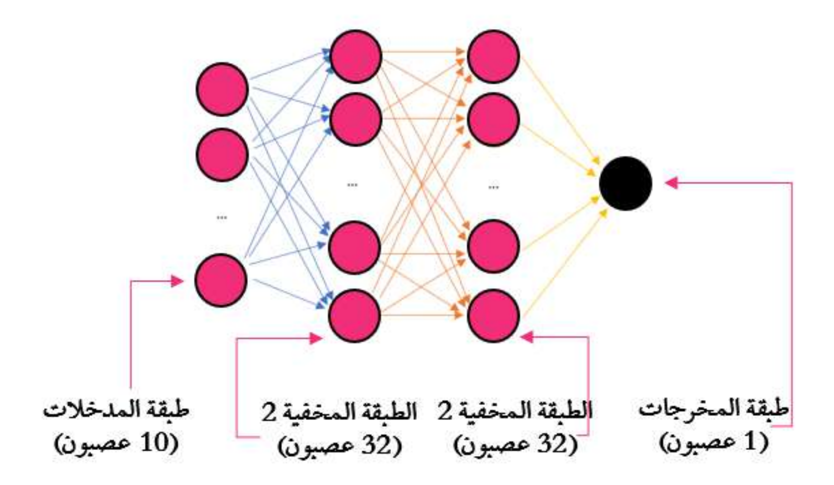
إذا كنت تريد أن ترى كيف تبدو المصفوفات لكل منها (أي ما هي أبعادها)، ما عليك سوى تشغيل الكود التالي:

```
print(X_train.shape, X_val.shape, X_test.shape, Y_train.shape,
Y_val.shape, Y_test.shape)
```

```
(1022, 10) (219, 10) (219, 10) (1022,) (219,) (219,)
```

كما ترى، تحتوي مجموعة التدريب على 1022 نقطة بيانات، بينما تحتوي كل مجموعة من مجموعات التحقق والاختبار على 219 نقطة بيانات. تحتوي المتغيرات X على 10 خصائص إدخال، بينما تحتوي المتغيرات Y على خاصية تنبؤ واحدة فقط.

حان الوقت الآن لبناء وتدريب شبكتنا العصبية الأولى. أول شيء يتعين علينا القيام به هو تكوين البُنية. لنفترض أننا نريد شبكة عصبية ذات بُنية على النحو التالي:



بعبارة أخرى، نريد هذه الطبقات:

- الطبقة المخفية 1: 32 خلية عصبية مع دالة التنشيط ReLU
- الطبقة المخفية 2: 32 خلية عصبية مع دالة التنشيط ReLU
- طبقة المخرجات: 1 خلية عصبية مع دالة التنشيط Sigmoid

الآن علينا وصف هذه الهيكلية في كيراس. سنستخدم النموذج المتسلسل (Sequential)، مما يعني أننا نحتاج فقط إلى وصف الطبقات العليا بالترتيب. أولاً، دعنا ندخل الكود المطلوب من Keras:

```
from keras.models import Sequential
from keras.layers import Dense
```

بعد ذلك، نحدد أن نموذجنا المتسلسل على النحو التالي:

```
model = Sequential([
    Dense(32, activation='relu', input_shape=(10,)),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid'),
])
```

تمامًا مثل الشكل السابق الذي رسمنا هيكله، يحدد مقتطف الكود أعلاه نفس الهيكلية. يمكن تفسير مقتطف الكود أعلاه على النحو التالي:

```
model = Sequential([...])
```

يوضح هذا الرمز أننا نقوم بتخزين نموذجنا في متغير النموذج ووصفه بالتسلسل (طبقة تلو طبقة) بين الأقواس.

```
Dense(32, activation='relu', input_shape=(10,))
```

في **الطبقة الأولى**، لدينا طبقة متصلة بالكامل بها 32 خلية عصبية، ودالة التنشيط ReLU وشكل الإدخال 10، لأن لدينا 10 خصائص إدخال. لاحظ أن كلمة "**Dense**" تشير إلى طبقة متصلة بالكامل.

```
Dense(32, activation='relu'),
```

الطبقة الثانية لدينا هي أيضًا طبقة متصلة بالكامل بها 32 خلية عصبية ودالة التنشيط ReLU. لاحظ أنه لا يتعين علينا وصف شكل الإدخال، حيث يمكن لـ Keras استنتاج ذلك من إخراج الطبقة الأولى.

```
Dense(1, activation='sigmoid'),
```

الطبقة الثالثة، أو طبقة الاخراج، هي طبقة متصلة بالكامل بها خلية عصبية واحدة ودالة التنشيط Sigmoid. كما ترون، تمكنا من برمجة هيكل النموذج الخاصة بنا.

الآن بعد أن حددنا المعمارية، نحتاج إلى إيجاد أفضل المعاملات لها. قبل بدء البرنامج التعليمي، نحتاج إلى تكوين النموذج من خلال ما يلي:

- **الخوارزمية التي تريد استخدامها لأداء التحسين.**
- **دالة الخطأ المستخدمة.**
- **المقاييس الأخرى التي تريد تتبعها بصرف النظر عن دالة الخطأ.**

لتكوين النموذج بهذه الإعدادات، نحتاج إلى استدعاء وظيفة model.compile ، على النحو التالي:

```
model.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

ضع الإعدادات التالية بين قوسين بعد `model.compile`:

```
optimizer='sgd',
```

يشير 'sgd' إلى الانحدار التصادفي العشوائي (يشير هنا إلى الانحدار التصادفي الصغير).

```
loss='binary_crossentropy',
```

بالنسبة للمخرجات التي تأخذ قيم 1 أو 0، يتم استخدام دالة الخطأ `'binary_crossentropy'`.

```
metrics=['accuracy']
```

أخيرًا، نريد تتبع الدقة جنبًا إلى جنب مع دالة الخطأ. الآن عندما نقوم بتشغيل هذه الخلية، نحن جاهزون للتدريب!

التدريب على الشبكةفي keras بسيط للغاية ويتطلب منا كتابة سطر واحد فقط من التعليمات البرمجية:

```
hist = model.fit(X_train, Y_train,
          batch_size=32, epochs=100,
          validation_data=(X_val, Y_val))
```

للقيام بذلك، نستخدم دالة "fit"، التي تلائم المعاملاتفي البيانات. نحتاج إلى تحديد البيانات التي نتدرب عليها والتي حددتها X_train وY_train. بعد ذلك، نحدد حجم المجموعة الفرعية (بواسطة المعامل batch_size) وطول الفترة الزمنية التي نريد تدريبها (epochs). أخيرًا، نحدد بيانات التحقق الخاصة بنا حتى يتمكن النموذج من إخبارنا بكيفية التعامل مع بيانات التحققفي أي مكان. تقوم هذه الدالة بإنشاء سجل نقوم بتخزينهفي متغير hist. سنستخدم هذا المتغير عندما نبدأفي الرسم والتوضيح. الآن، قم بتشغيل الخلية وشاهد البرنامج التدريبي الخاص بها! يجب أن تبدو مخرجاتك كما يلي:

```
Epoch 1/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6990 - accuracy: 0.3542 -
val_loss: 0.6974 - val_accuracy: 0.3699
Epoch 2/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6955 - accuracy: 0.4022 -
val_loss: 0.6943 - val_accuracy: 0.4110
Epoch 3/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6926 - accuracy: 0.4706 -
val_loss: 0.6915 - val_accuracy: 0.4703
Epoch 4/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6899 - accuracy: 0.5499 -
val_loss: 0.6889 - val_accuracy: 0.5616
Epoch 5/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6874 - accuracy: 0.6468 -
val_loss: 0.6864 - val_accuracy: 0.6758
Epoch 6/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6849 - accuracy: 0.7133 -
val_loss: 0.6842 - val_accuracy: 0.7123
Epoch 7/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6828 - accuracy: 0.7524 -
val_loss: 0.6821 - val_accuracy: 0.7489
Epoch 8/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6807 - accuracy: 0.7564 -
val_loss: 0.6801 - val_accuracy: 0.7717
Epoch 9/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6787 - accuracy: 0.7779 -
val_loss: 0.6781 - val_accuracy: 0.8037
Epoch 10/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6767 - accuracy: 0.8072 -
val_loss: 0.6761 - val_accuracy: 0.8128
Epoch 11/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6746 - accuracy: 0.8317 -
val_loss: 0.6740 - val_accuracy: 0.8219
Epoch 12/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6725 - accuracy: 0.8239 -
val_loss: 0.6717 - val_accuracy: 0.8265
.
.
.
.
Epoch 95/100
32/32 [==============================] - 0s 2ms/step - loss: 0.2931 - accuracy: 0.8865 -
val_loss: 0.3051 - val_accuracy: 0.9041
Epoch 96/100
32/32 [==============================] - 0s 2ms/step - loss: 0.2920 - accuracy: 0.8816 -
val_loss: 0.3043 - val_accuracy: 0.8995
Epoch 97/100
32/32 [==============================] - 0s 2ms/step - loss: 0.2911 - accuracy: 0.8855 -
val_loss: 0.3044 - val_accuracy: 0.9041
Epoch 98/100
32/32 [==============================] - 0s 2ms/step - loss: 0.2901 - accuracy: 0.8865 -
val_loss: 0.3030 - val_accuracy: 0.8995
Epoch 99/100
32/32 [==============================] - 0s 2ms/step - loss: 0.2896 - accuracy: 0.8806 -
val_loss: 0.3025 - val_accuracy: 0.8995
Epoch 100/100
32/32 [==============================] - 0s 2ms/step - loss: 0.2884 - accuracy: 0.8816 -
val_loss: 0.3017 - val_accuracy: 0.8995
```

الآن ترى أن النموذج يتم تدريبه! من خلال النظر إلى الأرقام، يجب أن تكون قادرًا على رؤية انخفاض الخطأ وزيادة الدقة بمرور الوقت.في هذه المرحلة، يمكنك اختبار الشبكة العصبية بمعاملات فائقة مختلفة. قم بتشغيل الخلايا مرة أخرى لترى كيف يتغير البرنامج التدريبي الخاص بك عندما تقوم بتغيير المعاملات الفائقة. بمجرد أن تشعر بالرضا عن نموذجك النهائي، يمكنك تقييمهفي مجموعة الاختبار. للعثور على دقة مجموعة الاختبار الخاصة بنا، نقوم بتشغيل مقتطف الشفرة البرمجية التالية:

```
model.evaluate(X_test, Y_test)[1]
```

سبب وجود الفهرس 1 بعد دالة التقييم هو أن الدالة ترجع الخطأ كعنصر أول والدقة كعنصر ثان. نظرًا لأنه دقيق بما يكفي لعرض المخرجات، يمكنك الوصول إليه بهذه الطريقة. نظرًا للتوزيع العشوائي **لمجموعة البيانات** وكذلك **تهيئة الأوزان**، ستكون الأرقام والرسوم البيانية مختلفة قليلاً في كل مرة نقوم فيها بتشغيل notebook الخاص بنا. ومع ذلك، إذا كنت قد اتبعت الهيكل الموضح أعلاه، فيجب أن تحصل على دقة اختبار تتراوح من 80 إلى 95٪! مثل الإخراج أدناه:

```
7/7 [==============================] - 0s 2ms/step - loss: 0.3281 - accuracy: 0.8584
0.8584474921226501
```

تهانينا! لقد استطعت ان تصمم وتدرب شبكتك العصبية الأولى.

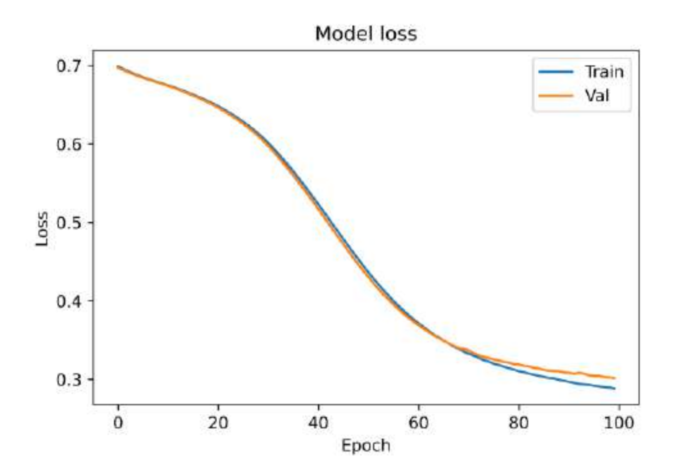
تحدثنافي الأقسام السابقة عن overfitting وبعض تقنيات التنظيم. الآن كيف نعرف أن نموذجنا بالفعل overfitting؟ ما يمكننا القيام به هو رسم الخطأ التدريبي وخطأ الاعتماد على عدد الدورات التي تم أخذها. لتوضيح ذلك، نستخدم حزمة matplotlib. كالعادة، علينا إدخال الكود الذي نريد استخدامه:

```
import matplotlib.pyplot as plt
```

ثم نريد توضيح خطأ التدريب `'loss'`وخطأ الاعتماد`'val_loss'`. للقيام بذلك، قم بتشغيل هذا المقطع من الكود:

```
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```

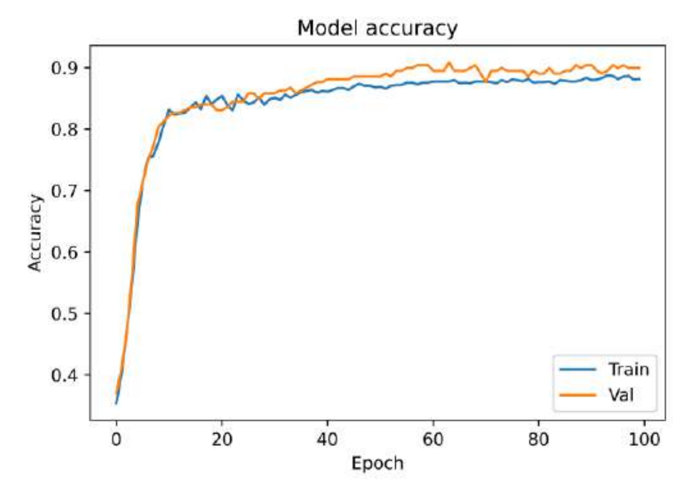
سنشرح كل سطر من مقتطف الشفرة أعلاه. يقول أول سطرين أننا نريد رسم خطأ التدريب `'loss'`وخطأ الاعتماد`'val_loss'`. يحدد السطر الثالث عنوان هذا المخطط: `Model loss`. يخبرنا الخطان الرابع والخامس ما يجب تسمية المحورين y و x على التوالي. يحتوي السطر السادس على وصف للمخطط الخاص بنا، وسيكون موقع الوصففي أعلى اليمين، ويخبر السطر السابع jupyter notebook لعرض المخطط. يجب أن تبدو مخرجاتك مثل هذا:



يمكننا أن نفعل الشيء نفسه لرسم دقة التدريب ودقة التحقق من الصحة باستخدام الكود التالي:

```
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='lower right')
plt.show()
```

يجب أن تحصل على رسم بياني يشبه هذا قليلاًفي الإخراج:



نظرًا لأن التحسينات التي أدخلت على نموذجنافي مجموعة التدريب تتوافق إلى حد ما مع تحسين مجموعة التحقق من الصحة، لا يبدو أن overfitting يمثل مشكلة كبيرةفي نموذجنا (ومع ذلك يمكن تحسينه عن طريق تحسين المعاملات الفائقة).

من أجل تنظيم الشبكة العصبية، دعنا نصمم شبكة عصبية تتلاءم بشكل مريح مع مجموعة التدريب. نسمي هذا النموذج model_2.

```
model_2 = Sequential([
    Dense(1000, activation='relu', input_shape=(10,)),
    Dense(1000, activation='relu'),
    Dense(1000, activation='relu'),
    Dense(1000, activation='relu'),
    Dense(1, activation='sigmoid'),
])

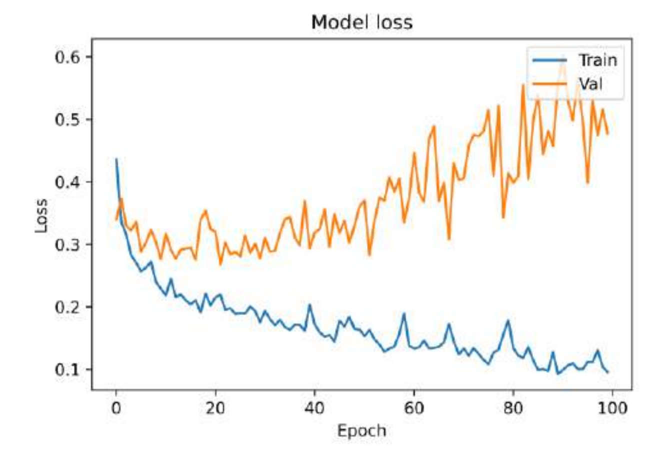
model_2.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

hist_2 = model_2.fit(X_train, Y_train,
          batch_size=32, epochs=100,
          validation_data=(X_val, Y_val))
```

هنا، نبني نموذجًا أكبر بكثير ونستخدم مُحسِّن Adam. يعد Adam أحد أكثر أدوات تحسين الأداء شيوعًا المستخدمةفي معماريات الشبكات العصبية، لأنه سريع ويتسببفي خطأ أقل. إذا قمنا بتشغيل هذا الكود ورسمنا مخططات الخطأ لـ hist_2 باستخدام الكود التالي (لاحظ أن الكود هو نفسه باستثناء أننا نستخدم hist_2 بدلاً من hist):

```
plt.plot(hist_2.history['loss'])
plt.plot(hist_2.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```

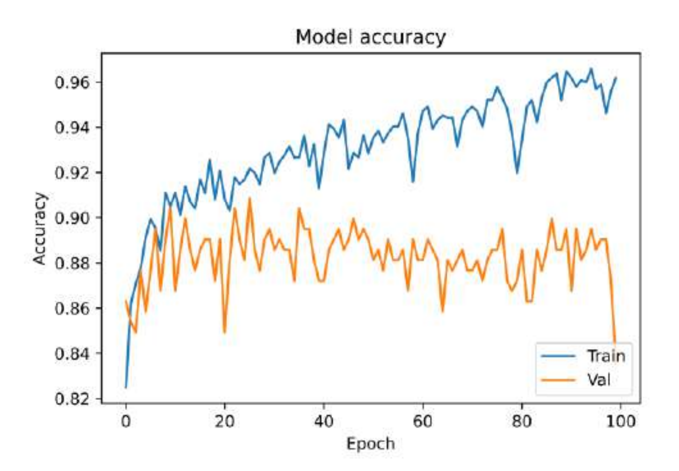
نحصل على رسم بياني على النحو التالي:



هذه علامة واضحة على overfitting. تتراجع الاخطاء التدريبية، لكن اخطاء الاعتماد أعلى بكثير من الاخطاء التدريبية وتتزايد. إذا رسمنا الدقة باستخدام الكود التالي:

```
plt.plot(hist_2.history['accuracy'])
plt.plot(hist_2.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='lower right')
plt.show()
```

يمكننا أيضًا أن نرى فرقًا أوضح بين الدقة التدريبية 'accuracy' ودقة الاعتماد 'val_accuracy':



الآن، دعنا نجرب بعض استراتيجياتنا لتقليل overfitting. في الأقسام السابقة، قدمنا عدة طرق لمنع overfitting. ومع ذلك، في هذا القسم، نستخدم فقط الحذف العشوائي dropout. أولاً، دعنا ندخل الكود الذي نحتاجه لإضافة الحذف التصادفي:

```
from keras.layers import Dropout
```

ثم نحدد نموذجنا الثالث على النحو التالي:

```
model_3 = Sequential([
    Dense(1000, activation='relu',  input_shape=(10,)),
    Dropout(0.5),
    Dense(1000, activation='relu'),
    Dropout(0.5),
    Dense(1000, activation='relu'),
    Dropout(0.5),
    Dense(1000, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid'),
])
```

هل يمكنك معرفة الفرق بين النموذج 3 والنموذج 2؟ هناك اختلاف رئيسي واحد:

لإضافة Dropout، أضفنا طبقة جديدة مثل هذه:

```
Dropout(0.5),
```

هذا يعني أن الخلايا العصبيةفي الطبقة السابقة لديها فرصة 0.5 للإزالة أثناء التدريب. دعنا نجمعها ونشغلها بنفس المعاملات مثل نموذج 2.

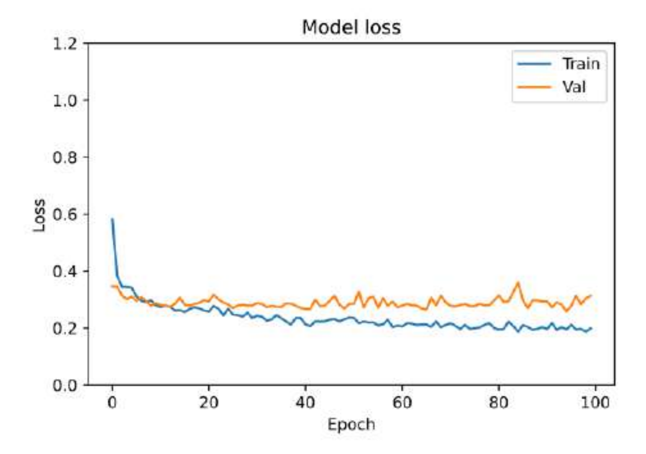
```
model_3.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

hist_3 = model_3.fit(X_train, Y_train,
          batch_size=32, epochs=100,
          validation_data=(X_val, Y_val))
```

الآن، دعنا نرسم مخططات الخطأ والدقة.

```
plt.plot(hist_3.history['loss'])
plt.plot(hist_3.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```

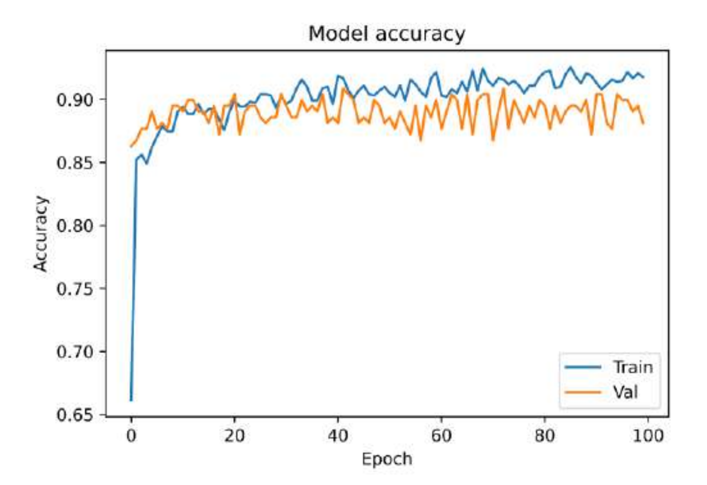
سوف نتلقى المخرجات على النحو التالي:



كما يمكن رؤيته، فإن خطأ التحقق من صحة النموذج 2 أكثر توافقاً مع خطا التدريب (ومع ذلك، لا يزال هذا النموذج غير مرغوب فيه والنموذج أكثر من اللازم). لنرسم الدقة بنفس الكود:

```
plt.plot(hist_3.history['accuracy'])
plt.plot(hist_3.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='lower right')
plt.show()
```

وسوف تظهر المخرجات على النحو التالي:



مقارنةً بالنموذج 2 ، قمنا بتقليل الضبط الزائد بشكل ملحوظ! هذه هي الطريقة التي نطبق بها تقنيات التنظيم لتقليل الضبط الزائد في مجموعة التدريب. كتمرين ، يمكنك تغيير المعاملات العليا ومقارنة النتائج.

خلاصة الفصل

- **الشبكات العصبية الاصطناعية هي نماذج حسابية تحاكي كيفية عمل الخلايا العصبية في الدماغ البشري.**
- **تحتوي كل شبكة عصبية اصطناعية على طبقة إدخال وطبقة إخراج وطبقة مخفية واحدة أو أكثر.**
- **يُطلق على أبسط نوع من نمذجة الخلايا العصبية اسم Perceptron ، والذي يمكن أن يحتوي على عدد كبير من المدخلات بمخرج واحد.**
- **يتمثل التقييد الرئيسي للشبكات العصبية الإدراكية Perceptron في عدم القدرة على تصنيف البيانات التي لا يمكن فصلها خطيًا.**
- **الهدف من عملية تعلم الشبكة العصبية هو العثور على مجموعة من قيم الاوزان التي تجعل ناتج الشبكة العصبية مطابقا قدر الإمكان مع القيم المستهدفة الفعلية.**
- **تقرر دالة التنشيط ما إذا كان يجب تنشيط الخلية العصبية ام لا.**

اختبار

10. ما هو تأثير اختيار معدل التعلم الصغير جداً أو الكبير جداً على عملية التعلم؟
11. صف ظاهرة تلاشي الانحدار؟
12. ما هي الميزات المرغوبة لدالة التنشيط؟
13. ما هي دالة التنشيط المستخدمة في طبقة الإخراج لمشاكل التصنيف الثنائي ومتعدد الفئات؟
14. ما الدور الذي يلعبه المحسن optimizer في عملية التعلم للشبكات العصبية؟

تمرين

قم بإنشاء شبكة عصبية ذات طبقتين مخفيتين لتصنيف مجموعة بيانات Iris. قم بتوضيح مخططات الدقة،والخطأ لمجموعة التدريب والتحقق من الصحة أثناء التدريب.

نصائح لتوسيع نطاق البيانات

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

نصائح لإدخال البيانات

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris['data']
y = iris['target']
```

4

الشبكة العصبية الالتفافية

اهداف التعليم:

- ماهي الشبكة العصبية الالتفافية؟
- التعرف على انواع الطبقات في الشبكات العصبية الالتفافية.
- تصنيف الصور باستخدام الشبكة العصبية الالتفافية.

المقدمة

في هذا الفصل، نقدم مفاهيم الشبكات العصبية الالتفافية (Convolutional neural network)، أو اختصارا CNN.تتضمن هذه المفاهيم المكونات الرئيسية للشبكة التي تشكل بُنية الشبكة العصبية الالتفافية. تعمل الشبكات العصبية الالتفافية بشكل جيد للغاية بالنسبة للبيانات غير المهيكلة مثل الصور. بعد التعرف الكامل على بنية الشبكات العصبية الالتفافية،في نهاية الفصل سنقوم بتنفيذ مثال عملي باستخدام الشبكات العصبية الالتفافيةفي keras.

الشبكات العصبية الالتفافية (CNN)

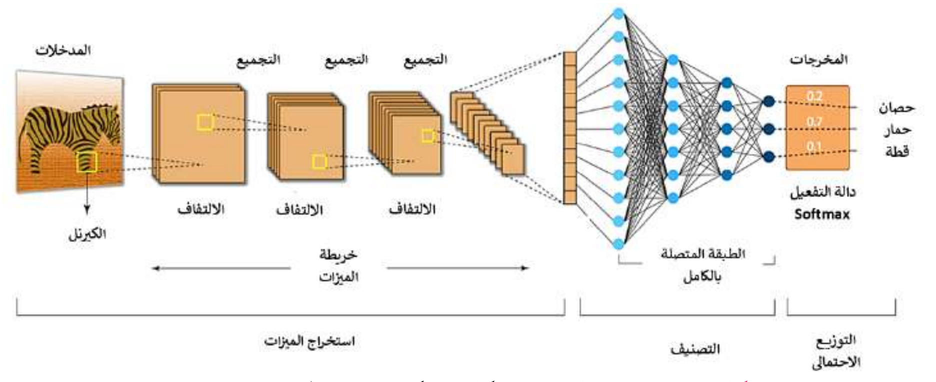
في الشبكات العصبية أمامية التغذية المتصلة بالكامل، يتم توصيل جميع العقد الخاصة بطبقة واحدة بجميع عقد الطبقة التالية. كل اتصال له وزن $w_{i,j}$ ،والذي يجب أن تتعلمه خوارزمية التعلم. لنفترض أن مدخلاتنا عبارة عن صورة 64 بكسل × 64 بكسل على مقياس رمادي. نظرًا لأنه يمكن تمثيل كل بكسل رمادي بقيمة 1، نقول إن حجم القناة هو 1. يمكن تمثيل هذه الصورة بـ $4096 = 1 \times 64 \times 64$ ((قناة × عمود ×صف). ومن ثم، فإن طبقة الإدخال للشبكة العصبية أمامية التغذية التي تعالج مثل هذه الصورة بها 4096 عقدة. افترض أن الطبقة التالية بها 500 عقدة. نظرًا لأن جميع العقدفي الطبقات التالية متصلة تمامًا، فسوف نقوم بحساب اوزان $2048000 = 500 \times 4096$ بين الإدخال والطبقة المخفية الأولى. ومع ذلك، بالنسبة للمشكلات المعقدة، نحتاج عادةً إلى عدة طبقات مخفيةفي الشبكة العصبية أمامية التغذية الخاصة بنا، لأن الشبكة العصبية أمامية التغذية البسيطة قد لا تكون قادرة على تعلم نموذج تعيين المدخلات إلى المخرجاتفي بيانات التدريب. يؤدي وجود طبقات مخفية متعددة إلى تفاقم مشكلة وجود عدد كبير جدًا من الأوزانفي شبكة التغذية الخاصة بنا ويجعل عملية التعلم أكثر صعوبة عن طريق زيادة حجم مساحة البحث. كما أنه يتيح المزيد من التدريب والوقت والموارد. بالإضافة إلى ذلك، فإن عددًا كبيرًا من المعاملات يزيد من ميل الشبكة إلى الضبط الزائد. لذلك، من أجل معالجة هذه المشكلات، تم تقديم الشبكات العصبية الالتفافية (CNN) كتطوير شائع جدًا للشبكات العصبية القياسية. الشبكات العصبية الالتفافية هي فئة من الشبكات العصبية أمامية التغذية التي تستخدم طبقات الالتفاف لتحليل المدخلات باستخدام طبولوجيا الشبكة، مثل الصور ومقاطع الفيديو. يعتمد اسم هذه الشبكات على دالة رياضية تسمى **الالتفاف**، والتي يستخدمونهافي بنيتهم. باختصار، الشبكات الالتفافية هي شبكات عصبية تستخدم الالتفاف بدلاً من ضرب المصفوفةفي طبقة واحدة على الأقل من طبقاتها.

ما يميز شبكة العصبية الالتفافية هو الطريقة التي يتم بها تنظيم الاتصالات بين الخلايا العصبية وبنية الطبقة المخفية الفريدة المستوحاة من آلية معالجة البيانات المرئية الخاصة بنا داخل القشرة

البصرية لدينا. على عكس الشبكات العصبية أمامية التغذية، الطبقاتفي CNN تنتظم بثلاثة أبعاد: العرض والارتفاع والعمق.

واحدة من أهم ميزات الشبكة الالتفافية التي يجب أن تضعهافي اعتبارك، بغض النظر عن عدد الطبقات الموجودةفي بُنيتها، هي أن البنية الكاملة لشبكة CNN تتكون من جزأين رئيسيين.:

- **استخراج الميزات(Features Extraction)**: في هذه الطبقة ، تقوم الشبكة بتنفيذ سلسلة من عوامل الالتفاف **(convolution)** والتجميع**(pooling)**. إذا أردنا تحديد صورة قطة في الصورة ، فهذا هو الجزء الذي يتم فيه التعرف على ميزات معينة مثل الأذنين والكفوف ولون فرو القطط. باختصار ، تتمثل مهمة هذه الطبقة في التعرف على الميزات المهمة في بكسلات الصورة. تتعلم الطبقات الأقرب إلى الإدخال التعرف على الخصائص البسيطة مثل الحواف وتدرجات الألوان ، بينما تدمج الطبقات الأعمق خصائص بسيطة مع خصائص أكثر تعقيداً.
- **التصنيف**: هنا تعمل الطبقة المتصلة بالكامل كمصنف بعد خطوة استخراج الميزة. تحدد هذه الطبقة الميزة الأكثر صلة بفئة معينة ، لذا فهي تتوقع فئة الصورة بناءً على الخصائص المستخرجة في الخطوات السابقة. يمكن رؤية معماريتها العامة في الشكل 4-1



شكل 4-1. لمحة عامة عن هيكل الشبكة العصبية الالتفافية

تتميز الشبكات العصبية الالتفافية بثلاث خصائص مميزة مقارنة بالشبكات العصبية الأخرى:

1. **المجالات المحلية الاستقبالية (local receptive fields)**: كل خلية عصبية في CNN مسؤولة عن منطقة محددة من بيانات الإدخال ، وهذا يسمح للخلايا العصبية بتعلم أنماط مثل الخطوط والحواف والتفاصيل الصغيرة التي تشكل الصورة. تسمى هذه المنطقة المحددة من الفضاء التي تتعرض لها خلية عصبية أو وحدة في بيانات الإدخال **المجال المحلي الاستقبالي**. يُعرَّف المجال الاستقبالي بحجم فلتر أحادي الطبقة في شبكة عصبية التفافية.

2. **مشاركة المعاملات (parameters sharing) و الارتباط المحلى (Local connectivity):** تحتوي كل طبقة التفافية على عدة فلاتر، وهذا معامل فائق محدد مسبقًا. يحتوي كل من هذه الفلاتر على عرض وارتفاع معدل يتوافق مع مجال الاستقبال المحلي للخلايا العصبية. تُنشئ الفلاتر التي تعمل على بيانات الإدخال خريطة المعالم عند إخراج الطبقة الالتفافية. مشاركة المعالم هي مشاركة الأوزان بواسطة جميع الخلايا العصبية في خريطة المعالم. الاتصال المحلي ، من ناحية أخرى ، هو مفهوم أن كل خلية عصبية مرتبطة فقط بمجموعة فرعية من الخلايا العصبية ، على عكس الشبكة العصبية الأمامية التي ترتبط فيها جميع الخلايا العصبية بشكل كامل. تساعد هذه الميزات في تقليل عدد المعاملات في جميع أنحاء النظام وجعل الحوسبة أكثر كفاءة.
3. **جمع العينات (sub-sampling) او التجميع (pooling):** غالبًا ما يأتي أخذ العينات الفرعية أو الدمج فورًا بعد طبقة الالتفاف على شبكة CNN. يتمثل دورها في خفض ناتج الطبقة الالتفافية على طول الأبعاد المكانية للارتفاع والعرض. تتمثل الوظيفة الرئيسية للتجميع في تقليل عدد المعاملات التي يجب أن تتعلمها الشبكة. تقلل هذه الميزة أيضًا من تأثير overfitting ، مما يؤدي إلى زيادة أداء الشبكة بشكل عام ودقتها.

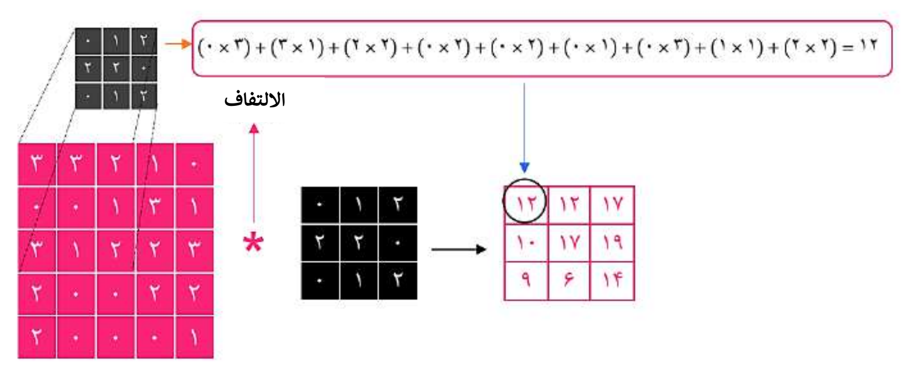
لعبت الشبكات الالتفافية دوراً مهمًا فى تاريخ التعلم العميق. إنها مثال مهم وناجح لفهمنا لدراسة الدماغ فى تطبيقات التعلم الآلى. كانت الشبكات العصبية الالتفافية من بين الشبكات العصبية الأولى التى تم استخدامها فى حل وتنفيذ تطبيقات الأعمال المهمة، وحتى يومنا هذا فهى فى طليعة تطبيقات الأعمال فى التعلم العميق.

تتمتع الاشبكات الالتفافية بقدرة كبيرة على العثور على أنماط مكررة واستخراج الميزات المحلية.

عامل الالتفاف

تنتمي الشبكات الالتفافية إلى مجموعة من الشبكات العصبية التي تأخذ الصورة كمدخلات، وتعرضها لمجموعة من الأوزان والتحيزات، وتستخرج الميزات، وتحصل على النتائج. تميل إلى تقليل حجم الصورة المدخلة باستخدام **الكيرنل (الفلتر)**، مما يجعل من السهل **استخراج الميزات** مقارنةً بالشبكة العصبية للتغذية. أساس الشبكة الالتفافية هو عامل الالتفاف.

الالتفاف ثنائي الأبعاد هوفي الأساس عملية بسيطة نسبيًا. تبدأ بكيرنل، وهي عبارة عن مصفوفة صغيرة من الأوزان. تنزلق الكيرنل على بيانات الإدخال ثنائية الأبعاد، وتضرب الكائنفي جزء الإدخال الموجود عليه حاليًا، ثم تلخص النتائجفي بكسل الإخراج (الشكل 4ـ2). تكرر الكيرنل هذه العملية لكل موقع تنزلق عليه (تلتف عليه)، وتحول مصفوفة ثنائية الأبعاد من الخصائص إلى مصفوفة خصائص ثنائية الأبعاد أخرى. خصائص الإخراج هيفي الأساس مجموع الاوزان لخصائص الإدخال التي تكون تقريبًافي نفس الموقع مثل بكسل الإخراجفي طبقة الإدخال.



شكل 4ـ2. عامل الالتفاف

في المثال أعلاه، خصائص الإدخال هي 25=5×5، وخصائص الإخراج 9=3×3. إذا كانت هذه طبقة قياسية متصلة بالكامل، فسنحصل على مصفوفة وزن برقم معامل 225=9×29، حيث تكون كل خاصية مخرجات هي الوزن الإجمالي لكل خاصية إدخال. تسمح لنا الالتفافات بإجراء هذا التحويل باستخدام 9 معاملات فقط.

تأثير الالتفاف هو التأكيد على حدود الأشكال المختلفة. يمكن تعديل الكيرنل المتغيرة لتلبية الاحتياجات الخاصة. ومع ذلك، بدلاً من محاولة القيام بذلك يدويًا، تقوم الشبكة الالتفافية العميقة بتفويض هذه المهام لعملية التعلم.

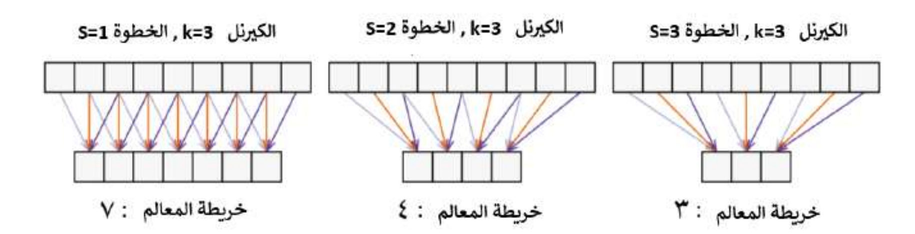
يؤدي التطبيق المتوازي للكيرنلات (الفلاتر) المختلفة إلى تداخلات معقدة يمكن أن تبسط استخراج الميزات المهمة حقًا للتصنيف. يتمثل الاختلاف الرئيسي بين الطبقة المتصلة تمامًا والطبقة الالتفافيةفي قدرة الطبقة الثانية على العمل مع هندسة موجودة تقوم بتشفير جميع العناصر اللازمة لتمييز كائن عن آخر. لا يمكن تعميم هذه العناصر على الفور، ولكنها تتطلب مزيداً من المعالجة لأداء الغموض. على سبيل المثال، العينان والأنف متطابقتان تقريبًا. كيف يمكن تقسيم الصورة بشكل صحيح؟ تأتي الإجابة بتحليل مزدوج: هناك اختلافات دقيقة يمكن اكتشافها باستخدام الفلاتر الدقيقة، والأهم من ذلك، أن الهندسة الكلية للكائنات تستند إلى علاقات داخلية ثابتة تقريبًا. على سبيل المثال، يجب أن تشكل العينان والأنف مثلثًا متساوي الأضلاع، لأن تناسق الوجه يعني نفس المسافة بين كل عين والأنف. يمكن القيام بذلك مسبقًا، مثل العديد من تقنيات معالجة الصور، أو يمكن تركه لعملية التعلم بفضل قوة التعلم العميق. نظرًا لأن دالة الخطأ وفئات المخرجات تتحكم ضمنيًافي الاختلافات، يمكن للشبكة الالتفافية العميقة معرفة ما هو مهم لتحقيق هدف محدد مع التخلص من جميع التفاصيل غير المجدية.

طبقة الالتفاف

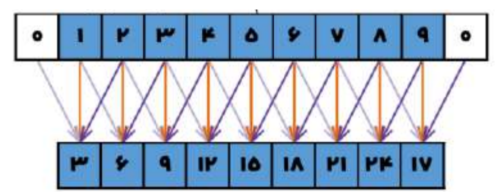
طبقة الالتفاف هي أهم لبنةفي بناء شبكة CNN. تحتوي هذه الطبقة على مجموعة من الفلاتر، تُعرف أيضًا باسم **الكيرنل (kernel)** أو كاشفات المعالم **(feature detectors)**، حيث يتم تطبيق كل فلتر على جميع مناطق بيانات الإدخال. بمعنى آخر، تتمثل المهمة الرئيسية لطبقة الالتفافي تحديد الخصائص الموجودةفي المناطق المحلية لصورة الإدخال، والتي تكون مشتركة لمجموعة البيانات بأكملها. ينتج عن التعرف على الميزة هذا إنتاج خريطة الميزات او المعالم عن طريق تطبيق عوامل التصفية. تطبق طبقة الالتفاف فلترا محليًا على صورة الإدخال. ينتج عن هذا تصنيف أفضل لوحدات البكسل المجاورة الأكثر ارتباطًافي نفس الصورة. بمعنى آخر، يمكن أن ترتبط وحدات البكسل الخاصة بالصور المدخلة ببعضها البعض. على سبيل المثال،في صور الوجه، يكون الأنف دائمًا بين العينين والفم. عندما نطبق المرشح على مجموعة فرعية من الصورة، فإننا نستخرج بعض الخصائص المحلية. يشار إلى هذه الطبقة أيضًا باسم طبقة **استخراج الميزات او المعالم**. لأنه يتم استخراج خصائص الصورةفي هذه الطبقة.

تحتوي كل طبقة التفاف على مجموعة محددة من المعاملات الفائقة، كل منها يحدد عدد الاتصالات وحجم الإخراج لخرائط الميزات:

- **حجم الكيرنل:** يصف الحجم الأساسي K (يسمى أحيانًا **حجم الفلتر**) الحقل الاستقبالي الذي ينطبق على جميع مواقع الإدخال. تسمح زيادة هذا المعامل لطبقة الالتفاف باستقبال المزيد من المعلومات المكانية ، مع زيادة عدد أوزان الشبكة في نفس الوقت.
- **عدد الكيرنل:** يتوافق عدد الكيرنلات بشكل مباشر مع عدد المعاملات القابلة للتعلم وعمق D لحجم إخراج طبقة الالتفاف. مثلما ينتج كل كيرنل خريطة ميزات مخرجات منفصلة ، فإن الكيرنلات D تنتج خريطة ميزات مخرجات بعمق D.
- **الخطوة:** يمكن اعتبار الالتفاف تراكميًا عن طريق "انزلاق" الكيرنل على حجم الإدخال. ومع ذلك ، لا يجب أن يحدث "الانزلاق" على مسافة بكسل واحد في كل مرة ، وهو ما تصفه **الخطوة (Stride)**. تحدد الخطوة S عدد وحدات البكسل التي تنقلها الكيرنل بين كل حساب لخاصية الإخراج. تنتج الخطوات الأكبر حجمًا خرائط ميزات مخرجات أصغر لأنه يتم إجراء عدد أقل من العمليات الحسابية. يظهر هذا المفهوم في الشكل أدناه:



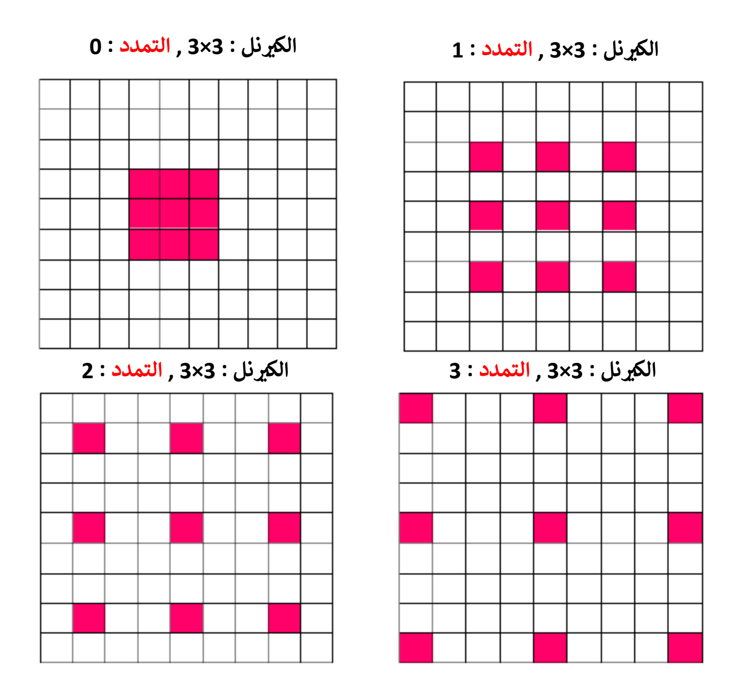
- **الطبقات الصفرية:** نظرًا لتشغيل عملية الالتفاف ، تُستخدم الطبقات الصفرية (Padding) او الحشو للتحكم في الأبعاد بعد تطبيق مرشحات أكبر من 1 * 1 ولمنع فقدان المعلومات في الحواشي. بمعنى آخر ، غالبًا ما يتم استخدام الطبقة الصفرية للحفاظ على الأبعاد المكانية لطبقات الإدخال والإخراج كما هي. عن طريق إضافة مدخلات صفرية حول المحيط ، يمكن تجنب انكماش الأبعاد المكانية عند الالتفاف. تعد قيمة الأصفار المضافة على كل جانب لكل بُعد مكاني معامل فائق إضافي P. يظهر مثال على التقسيم الصفري في الشكل أدناه:



- **التمدد (Dilation):** التمدد او الفتحة d التي تم إدخالها مؤخرًا هي معامل فائق أخرى يتيح لطبقة الالتفاف أن يكون لها مجال استقبال أكثر كفاءة من الإدخال ، مع الحفاظ على حجم الكيرنل ثابتًا. يتم الحصول على ذلك عن طريق إدخال المسافة d بين كل خلية من الكيرنل. يستخدم الالتفاف القياسي ببساطة التمدد 0. ومن ثم فإن لها كيرنل مستمرة. من خلال زيادة المساحة ، يمكن لطبقة الالتفاف أن تشغل مساحة أكبر من الإدخال مع الحفاظ على ثبات استهلاك الذاكرة. يظهر مفهوم الالتفافات التمددية ، التي تسمى أحيانًا **الالتفافات الأذينية (atrous convolutions)**، في الشكل 3ـ4ـ4 مع تمددات مختلفة.

وفقًا لحجم الإدخال W ، حجم الكيرنل لـ K ، الخطوة S ، التمدد d و P طبقة الصفرية، يتم حساب حجم الإخراج الناتج على النحو التالي:

$$W_o = \left\lfloor \frac{W + 2P - K - (K-1)(d-1)}{S} \right\rfloor + 1.$$



شكل 4ـ3. التمدد على مدخلات ثنائية الأبعاد بأحجام مختلفة.

استخدام الالتفاف له ثلاث مزايا مهمة. أولاً، عادةً ما يكون للشبكات العصبية الالتفافية **تفاعلات متفرقة (Sparse interactions)**. تستخدم الشبكات العصبية أمامية التغذية مصفوفة من المعاملات التي تصف العلاقة بين وحدات الإدخال والإخراج. هذا يعني أن كل وحدة إخراج متصلة بكل وحدة إدخال. ومع ذلك، فإن الشبكات العصبية الالتفافية لها تفاعلات متفرقة يتم تحقيقه عن طريق تقليص الكيرنل من المدخلات. على سبيل المثال، يمكن أن تحتوي الصورة على ملايين أو آلاف وحدات البكسل، ولكن أثناء معالجتها باستخدام الكيرنل، يمكننا تحديد المعلومات المفيدة التي تتكون من عشرات أو مئات من وحدات البكسل. هذا يعني أنه يتعين علينا تخزين عدد أقل من المعاملات التي لا تقلل من الحاجة إلى الذاكرة فحسب، بل تعمل أيضًا على تحسين الأداء الإحصائي للنموذج. ثانيًا، تستخدم الشبكات العصبية الالتفافية **مشاركة المعاملات**. أي أنهم يعيدون استخدام نفس المعاملات للعديد من الدوال. تعطي المعاملات المشتركة أيضًا الميزة الرئيسية الأخيرة، **التقارب (Equivariance)**. التقارب يعني أنه في حالة ازاحة المدخلات، يتم إزاحة المخرجات بنفس الطريقة. هذه الميزة ضرورية لمعالجة البيانات ثنائية الأبعاد، لأنه إذا تم نقل صورة أو جزء منها إلى موقع آخرفي الصورة، فسيكون لها نفس العرض.

تربط الشبكات العصبية للتغذية كل خلية عصبية إدخال بجميع الخلايا العصبية فى الطبقة التالية، والتى تسمى عملية الاتصال الكاملة، ومع ذلك، تتطلب هذه الطريقة

حساب الأوزان الإضافية بطريقة تؤثر بشكل كبير على سرعة تدريب النموذج. بدلاً من إجراء اتصال كامل، تستخدم CNN اتصالًا جزئيًا، مما يعني أن كل خلية عصبية متصلة فقط بمنطقة من طبقة الإدخال المعروفة باسم المجال المحلي للخلايا العصبية المخفية. لذلك، تحتوي شبكة CNN على معاملات أقل من الشبكة العصبية امامية التغذية وبالتالي تتمتع بسرعة تدريب أسرع.

في طبقات الالتفاف، يتم الحصول على خرائط الميزات من بيانات الإدخال باستخدام عامل الالتفاف.

طبقة الالتفاف في keras

لإنشاء طبقة التفاف في Keras، يجب عليك أولاً إدخال الوحدات المطلوبة على النحو التالي:

```
from keras.layers import Conv2D
```

يمكنك بعد ذلك إنشاء طبقة الالتفاف باستخدام التنسيق التالي:

```
Conv2D (filters, kernel_size, strides, padding, activation='relu',
input_shape)
```

يجب عليك إدخال الوسيطات التالية:

- filters: تعداد الفلاتر
- kernel_size: رقم يشير إلى ارتفاع وعرض نافذة الالتفاف (حجم الكيرنل).
- strides: خطوة الالتفاف. إذا لم تحدد أي شيء ، فسيتم تعيينه على واحد افتراضيًا.
- padding: valid او same
- activation: عادة ما يتم استخدام دالة التنشيط relu.

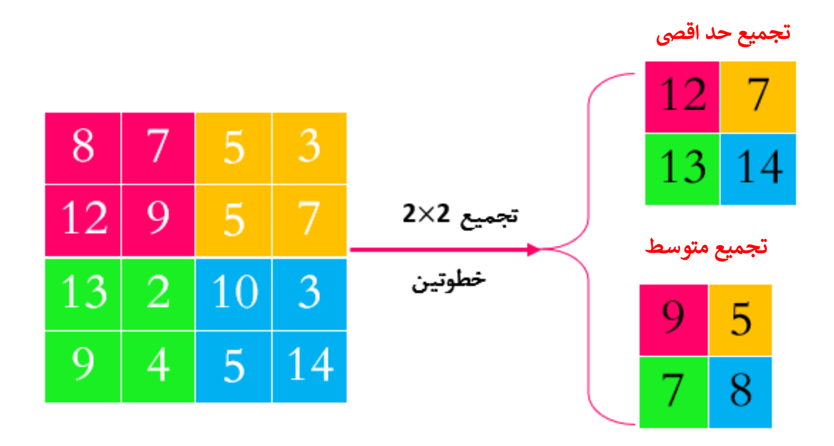
عند استخدام طبقة الالتفاف الخاصة بك كطبقة أولى في النموذج، يجب عليك إدخال وسيطة input_shape إضافية. هذه مجموعة تحدد ارتفاع المدخلات وعرضها وعمقها (على التوالي).

تأكد من عدم تضمين وسيطة input_shape إذا لم تكن طبقة الالتفاف هي الطبقة الأولى في شبكتك.

طبقة التجميع (الدمج)

تتمثل إحدى مزايا طبقات الالتفاف في أنها تقلل عدد المعاملات المطلوبة وتحسن الأداء وتقلل من الضبط الزائد. بعد عملية الالتفاف، غالبًا ما يتم إجراء عملية أخرى: **الدمج او التجميع (Pooling)**. تساعد طبقة التجميع على تقليل مقدار قوة الحوسبة المطلوبة لمعالجة البيانات وهي مسؤولة عن تقليل الأبعاد. بمساعدة تقليل الأبعاد، يتم تقليل مقدار قوة المعالجة المطلوبة لمعالجة مجموعة البيانات. يمكن تقسيم التجميع إلى نوعين: **تجميع الحد الأقصى (maximum pooling)** **تجميع المتوسط (average pooling)**. النوع الأكثر شيوعًا للتجميع

هو الحد الأقصى من التجميع والشبكات (2×2)في كل جزء واختيار الخلايا العصبية مع أقصى قدر من التنشيطفي كل شبكة واستبعاد الباقي. من الواضح أن مثل هذه العملية تزيل 75٪ من الخلايا العصبية وتحافظ فقط على الخلايا العصبية التي تلعب الدور الأكبر.في المقابل،في التجميع المعتمد على المتوسط ، يتم حساب متوسط قيمة الكيرنل (الشكل 3ـ4).



شكل 4ـ4. الحد الأقصى للتجميع ومتوسط التجميع

لكل طبقة دمج، هناك معاملان : **حجم الخلية وخطوتها**، على غرار معاملات الخطوة والطبقات في طبقات الالتفاف. الخيار الشائع هو اختيار حجم الخلية 2 والخطوة 2. ومع ذلك، فإن اختيار حجم الخلية 3 والخطوة 2 ليس من غير المألوف. تجدر الإشارة إلى أنه إذا كان حجم الخلية كبيرًا جدًا، فقد تتجاهل طبقة الدمج الكثير من المعلومات وقد لا تكون مفيدة.

وتجدر الإشارة إلى أنه مثل كيفية استخدام دوال التنشيط المختلفة، يمكننا أيضًا استخدام عوامل تجميع مختلفة. ومع ذلك، يعد استخدام تجميع الحد الأقصى أحد أكثر عوامل التشغيل شيوعًا، ولكن تجميع المتوسط ليس نادرًا. من الناحية العملية، غالبًا ما يعمل الحد الأقصى من التكامل بشكل أفضل، لأنه يحافظ على الهياكل الأكثر صلة في الصورة.

لاحظ أن طبقات الدمج لا تضيف أي معاملات جديدة، لأنها ببساطة تستخرج القيم (مثل الحد الأقصى) دون الحاجة إلى وزن إضافي أو انحياز.

تصنيف الصور مع الشبكة الالتفافية في keras

لتصنيف الصور، نحتاج أولاً إلى مجموعة من البيانات والعلامات لكل صورة. لحسن الحظ، لا يتعين علينا تجريف الويب (**scrape**) يدويًا بحثًا عن الصور ووضع علامة عليها بأنفسنا، حيث

توجد العديد من مجموعات البيانات **(Dataset)** القياسية التي يمكننا استخدامها.في هذا المثال، سوف نستخدم مجموعة بيانات CIFAR-10. تفاصيل مجموعة البيانات كما يلي:

- **أبعاد الصورة:** صورة مصغرة 32×32 بكسل.
- **العلامات (التسميات):** 10 علامات (مخرجات) تشمل: طائرة ، سيارة ، طائر ، قطة ، غزال ، كلب ، ضفدع ، حصان ، سفينة وشاحنة.
- **حجم مجموعة البيانات:** 60.000 صورة مقسمة إلى 50000 بيانات للتدريب و 10000 بيانات للاختبار.

أول شيء يتعين علينا القيام به هو استيراد مجموعة بيانات الصورة. نفعل هذا مع Keras عن طريق تشغيل الكود التالي على jupyter notebook:

```
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 2s 0us/step
170508288/170498071 [==============================] - 2s 0us/step
```

يتم الآن تخزين البيانات التي نحتاجهافي المصفوفات المقابلة (x_train, y_train) و (x_test, y_test). دعنا نلقي نظرة فاحصة على مجموعة البيانات. دعونا نرى كيف تبدو مصفوفة خصائص الإدخال لدينا:

```
print('x_train shape:', x_train.shape)
```

```
x_train shape: (50000, 32, 32, 3)
```

يخبرنا شكل المصفوفة أن مجموعة بيانات x_train تحتوي على ما يلي:

- 50000 صورة
- ارتفاع 32 بكسل
- عرض 32 بكسل
- عمق 3 بكسل (مرتبط باللون الأحمر والأخضر والأزرق)

دعونا نرى كيف تبدو مصفوفة العلامات (المخرجات):

```
print('y_train shape:', y_train.shape)
```

```
y_train shape: (50000, 1)
```

هذا يعني أن هناك رقمًا (مرتبطًا بالعلامة) لكل صورة من 50000 صورة. الآن، دعنا نحاول رؤية مثال على صورة وعلامتها لفهم أفضل:

```
print(x_train[0])
```

```
[[[ 59  62  63]
  [ 43  46  45]
  [ 50  48  43]
  ...
```

```
  [158 132 108]
  [152 125 102]
  [148 124 103]]

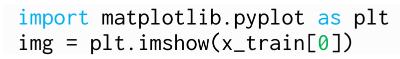
 [[ 16  20  20]
  [  0   0   0]
  [ 18   8   0]
  ...
  [123  88  55]
  [119  83  50]
  [122  87  57]]

 [[ 25  24  21]
  [ 16   7   0]
  [ 49  27   8]
  ...
  [118  84  50]
  [120  84  50]
  [109  73  42]]

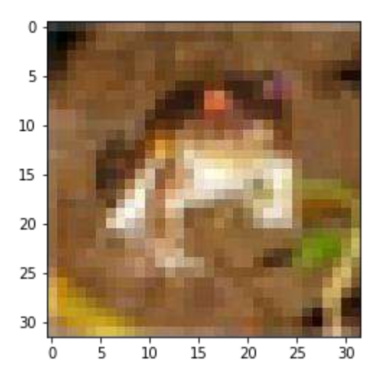
 ...

 [[208 170  96]
  [201 153  34]
  [198 161  26]
  ...
  [216 184 140]
  [151 118  84]
  [123  92  72]]]
```

بينما يرى الكمبيوتر الصورة بهذه الطريقة، فإنها ليست مفيدة جداً لنا. لذلك دعونا نوضح صورة x_train[0] [0] باستخدام حزمة matplotlib:



```
import matplotlib.pyplot as plt
img = plt.imshow(x_train[0])
```



plt.imshow هي دالة تعرض قيم البكسل المرقمةفي x_train[0] كصورة فعلية. الصورة الموضحة أعلاه منقطة للغاية، وذلك لأن حجم الصورة هو 32 × 32 بكسل وهو صغير جداً. الآن دعونا نرى ما هي علامة هذه الصورةفي مجموعة البيانات الخاصة بنا:

```
print('The label is:', y_train[0])
```

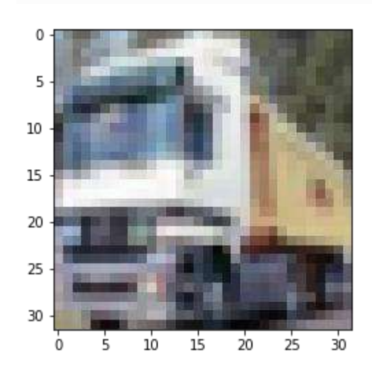
```
The label is: [6]
```

نرى أن التسمية "6". يتم ترتيب تحويل الأرقام إلى تسميات بناءً على أحرف الأبجدية الإنجليزية على النحو التالي:

التسمية	الرقم
مطار	0
سيارة	1
عصفور	2
قطة	3
غزال	4
كلب	5
ضفدع	6
حصان	7
سفينة	8
شاحنة	9

لذلك، نرى من الجدول أن الصورة أعلاه وصفت بأنها صورة ضفدع (التسمية 6). لنلقِ نظرة على مثال آخر للصورة عن طريق تغيير الفهرس إلى 1 (الصورة الثانية في مجموعة البيانات الخاصة بنا) بدلاً من 0 (الصورة الأولى في مجموعة البيانات الخاصة بنا):

```
img = plt.imshow(x_train[1])
```



دعونا نظهر العلامة كذلك:

```
print('The label is:', y_train[1])
```

```
The label is: [9]
```

باستخدام الجدول السابق، نرى أن هذه الصورة معلمة على أنها شاحنة.

الآن بعد أن راجعنا مجموعة البيانات الخاصة بنا، نحتاج إلى معالجتها. الملاحظة الأولى التي نجريها هي أن تسمياتنا ليست مفيدة جداً كرقم فئة. هذا لأن الفصول ليست بالترتيب. لتوضيح هذه النقطة، دعنا نعطي مثالاً. ماذا يحدث إذا لم تتمكن شبكتنا العصبية من تحديد ما إذا كانت الصورة سيارة (التسمية: 1) أو شاحنة (التسمية: 9). هل يجب أن نعتبر المتوسط ونتوقعه ككلب (التسمية: 5)؟ بالتأكيد مثل هذا الشيء لا معنى له.

في الفصل السابق ، أنشأنا شبكتنا العصبية الأولى للتنبؤ بأسعار المنازل باستخدام Keras ، قد تتساءل عن سبب تمكننا من استخدام علامتي [0] و [1] هناك. هذا بسبب وجود فئتين فقط ويمكننا تفسير ناتج الشبكة العصبية على أنه احتمال. بمعنى ، إذا كان ناتج الشبكة العصبية 0.6 ، فهذا يعني أنها تعتقد أنها أعلى بنسبة 60٪ من متوسط سعر المنزل. ومع ذلك ، لا يعمل هذا في تكوينات متعددة الطبقات مثل هذا المثال ، حيث يمكن أن تنتمي الصورة إلى واحد من 10 فئات مختلفة.

ما نريده حقًا هو احتمال كل فئة من الفئات العشر المختلفة. من أجل ذلك ، نحتاج إلى 10 عصبونات ناتجة في شبكتنا العصبية. نظرًا لأن لدينا 10 خلايا عصبية ناتجة ، يجب أن تتطابق تسمياتنا أيضًا. للقيام بذلك ، نقوم بتحويل التسمية إلى مجموعة من 10 أرقام ، كل منها يشير إلى ما إذا كانت الصورة تنتمي إلى تلك الفئة أم لا. لذلك إذا كانت الصورة تنتمي إلى الفئة الأولى، فسيكون الرقم الأول في هذه الفئة 1 وجميع الأرقام الأخرى في هذه الفئة ستكون 0. يسمى هذا **ترميز (one-hot)** ، ويكون جدول التحويل لهذا المثال على النحو التالي:

الرقم	التسمية	ترميز one-hot
0	مطار	[1000000000]
1	سيارة	[0100000000]
2	عصفور	[0010000000]
3	قطة	[0001000000]
4	غزال	[0000100000]
5	كلب	[0000010000]
6	ضفدع	[0000001000]
7	حصان	[0000000100]
8	سفينة	[0000000010]
9	شاحنة	[0000000001]

للقيام بهذا التحويل ، استخدم Keras مرة أخرى:

```
from keras.utils import np_utils
y_train_one_hot = keras.utils.np_utils.to_categorical(y_train, 10)
y_test_one_hot = keras.utils.np_utils.to_categorical(y_test, 10)
```

يعني السطر`y_train_one_hot = keras.utils.np_utils.to_categorical(y_train, 10)` أننا نأخذ المصفوفة الأصلية برقم `y_train` فقط ونحولها إلى `y_train_one_hot` وذلك باستخدام ترميز one_hot. الرقم 10 مطلوب كمعامل لأنه يجب عليك إخبار الدالة بعدد الفئات الموجودة.

الآن ، لنفترض أننا نريد أن نرى كيف تبدو تسمية الصورة الثانية (الشاحنة ذات العلامة: 9) في هذا الكود:

```
print('The one hot label is:', y_train_one_hot[1])
```

```
The one hot label is: [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```

الآن بعد أن عالجنا علامات (y) الخاصة بنا ، قد نرغب في معالجة صورتنا (x) أيضًا. الخطوة الشائعة التي نتخذها هي ترك القيم بين 0 و 1 ، مما يساعد على تدريب شبكتنا العصبية. نظرًا لأن قيم البكسل لدينا تأخذ بالفعل قيمًا بين 0 و 255 ، فنحن نحتاج ببساطة إلى تقسيمها على 255:

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train = x_train / 255
x_test = x_test / 255
```

عمليا، ما نقوم به هو تحويل النوع إلى "float32" ، وهو نوع بيانات يمكنه تخزين القيم في الكسور العشرية. ثم نقسم كل خلية على 255. إذا كنت ترغب في ذلك ، يمكنك إلقاء نظرة على قيم المصفوفات للصورة التدريبية الأولى عن طريق تنفيذ الخلية:

```
x_train[0]
```

```
array([[[0.23137255, 0.24313726, 0.24705882],
        [0.16862746, 0.18039216, 0.1764706 ],
        [0.19607843, 0.1882353 , 0.16862746],
        ...,
        [0.61960787, 0.5176471 , 0.42352942],
        [0.59607846, 0.49019608, 0.4       ],
        [0.5803922 , 0.4862745 , 0.40392157]],

       [[0.0627451 , 0.07843138, 0.07843138],
        [0.        , 0.        , 0.        ],
        [0.07058824, 0.03137255, 0.        ],
        ...,
        [0.48235294, 0.34509805, 0.21568628],
        [0.46666667, 0.3254902 , 0.19607843],
        [0.47843137, 0.34117648, 0.22352941]],

       [[0.09803922, 0.09411765, 0.08235294],
        [0.0627451 , 0.02745098, 0.        ],
        [0.19215687, 0.10588235, 0.03137255],
        ...,
        [0.4627451 , 0.32941177, 0.19607843],
        [0.47058824, 0.32941177, 0.19607843],
        [0.42745098, 0.28627452, 0.16470589]],

       ...,

       [[0.8156863 , 0.6666667 , 0.3764706 ],
        [0.7882353 , 0.6       , 0.13333334],
        [0.7764706 , 0.6313726 , 0.10196079],
        ...,
        [0.627451  , 0.52156866, 0.27450982],
        [0.21960784, 0.12156863, 0.02745098],
        [0.20784314, 0.13333334, 0.07843138]],

       [[0.7058824 , 0.54509807, 0.3764706 ],
```

```
[0.6784314 , 0.48235294, 0.16470589],
[0.7294118 , 0.5647059 , 0.11764706],
...,
[0.72156864, 0.5803922 , 0.36862746],
[0.38039216, 0.24313726, 0.13333334],
[0.3254902 , 0.20784314, 0.13333334]],

[[0.69411767, 0.5647059 , 0.45490196],
 [0.65882355, 0.5058824 , 0.36862746],
 [0.7019608 , 0.5568628 , 0.34117648],
 ...,
 [0.84705883, 0.72156864, 0.54901963],
 [0.5921569 , 0.4627451 , 0.32941177],
 [0.48235294, 0.36078432, 0.28235295]]], dtype=float32)
```

حتى الآن لدينا مجموعة تدريب واحدة ومجموعة اختبار واحدة. على عكس المثال الوارد في الفصل السابق ، لا نقوم بتقسيم مجموعة التحقق الخاصة بنا مقدمًا ، حيث يوجد اختصار لهذا سنقوم بتقديمه لاحقًا.

على غرار المثال السابق ، يجب علينا أولاً تحديد معمارية النموذج الخاصة بنا. معمارية CNN التي سنبنيها هي كما يلي:



وقيم معاملات المعمارية المذكورة أعلاه تتلخص على النحو التالي:

- Conv Layer (32 Filter size 3×3)
- Conv Layer (32 Filter size 3×3)
- Max Pool Layer (Filter size 2×2)
- Dropout Layer (Prob of dropout 0.25)
- Conv Layer (64 Filter size 3×3)
- Conv Layer (64 Filter size 3×3)
- Max Pool Layer (Filter size 2×2)
- Dropout Layer (Prob of dropout 0.25)
- FC Layer (512 neurons)
- Dropout Layer (Prob of dropout 0.5)
- FC Layer, Softmax (10 neurons)

تحتوي هذه المعمارية على الكثير من الطبقات (أكثر مما رأيناه من قبل) ، لكنها كلها مبنية من مفاهيم رأيناها من قبل. ومع ذلك ، يتم إنشاء كل طبقة بسطر واحد فقط من التعليمات البرمجية

ولا داعي للقلق! لاحظ أن وظيفة softmax تقوم ببساطة بتحويل إخراج الطبقة السابقة إلى توزيعات محتملة ، وهو ما نريده لمشكلة التصنيف الخاصة بنا.

لبرمجة هذا ، سنستخدم نموذج Keras المتسلسل. ومع ذلك ، نظرًا لأن لدينا العديد من الطبقات في نموذجنا ، فإننا نقدم طريقة جديدة للتسلسل. نمر عبر الكود سطرًا سطرًا حتى تتمكن من متابعة ما نفعله بالضبط. أولاً نقوم بإدخال بعض التعليمات البرمجية التي نحتاجها:

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
```

ثم نقوم بإنشاء نموذج تسلسلي فارغ:

```
model = Sequential()
```

نضيف طبقة واحدة إلى هذا النموذج الفارغ في كل مرة. الطبقة الأولى (إذا كنت تتذكر من الشكل السابق) هي طبقة الالتفاف مع حجم فلتر 3×3 ، وحجم الخطوة 1 وعمق 32. الطبقة هي نفسها والمنشط هو "relu" (هذان التكوينان ينطبقان على جميع طبقات CNN). ومع ذلك، دعنا نحدد الطبقة الأولى بالكود التالي:

```
model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
input_shape=(32,32,3)))
```

ما يفعله هذا الكود هو إضافة هذه الطبقة إلى نموذجنا المتسلسل الفارغ باستخدام الدالة `model.add()` . الرقم الأول ، 32 ، يشير إلى عدد الفلاتر. الزوج التالي من الأرقام (3،3) يشير إلى عرض وحجم الفلتر. ثم نحدد التنشيط وهو `'relu'` والطبقة `'same'`. لاحظ أننا لم نحدد خطوة ، هذا لأن `stride=1` هي إعداد افتراضي ولسنا بحاجة إلى تحديده ما لم نرغب في تغييره. إذا كنت تتذكر ، فنحن بحاجة أيضًا إلى تحديد حجم الإدخال لطبقتنا الأولى. لا تحتوي الطبقات اللاحقة على هذه المواصفات ، حيث يمكنها استنتاج حجم الإدخال من حجم الإخراج للطبقة السابقة. تبدو الطبقة الثانية في الكود على هذا النحو (لا نحتاج إلى تحديد حجم الإدخال):

```
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
```

الطبقة التالية هي طبقة حد اقصى من التجميع بحجم تجميع 2 × 2 وخطوتين. الحجم الافتراضي لطبقة اقصى حد من التجميع هو 2 × 2 ، لذلك لا نحتاج إلى تحديد هذه الخطوة:

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

أخيرًا ، نضيف طبقة الحذف العشوائي مع احتمال 0.25 لمنع overfitting:

```
model.add(Dropout(0.25))
```

الآن قمنا بإنشاء أول أربع طبقات مع الكود. تبدو الطبقات الأربع التالية متشابهة حقًا:

```
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Dropout(0.25))
```

أخيرًا ، نحتاج إلى برمجة الطبقة المتصلة بالكامل ، وهو ما يشبه ما فعلناه في المثال في الفصل السابق. ومع ذلك ، في هذه المرحلة ، يتم ترتيب الخلايا العصبية لدينا في مكعب بدلاً من صف. لوضع هذا الشكل الشبيه بالمكعب من الخلايا العصبية في صف واحد ، يجب أن نقوم أولاً بتسويته. نقوم بذلك عن طريق إضافة طبقة Flatten:

```
model.add(Flatten())
```

الآن ، نحن بحاجة إلى إنشاء طبقة متصلة بالكامل بها 512 خلية عصبية ودالة التنشيط relu:

```
model.add(Dense(512, activation='relu'))
```

ثم نضيف حذفًا عشوائيًا آخر باحتمال 0.5:

```
model.add(Dropout(0.5))
```

أخيرًا ، قمنا بإنشاء طبقة متصلة بالكامل بها 10 خلايا عصبية ودالة التنشيط softmax:

```
model.add(Dense(10, activation='softmax'))
```

تم الانتهاء الآن من بناء معماريتنا. الآن ، لرؤية ملخص للمعمارية الكاملة ، نقوم بتشغيل الكود التالي:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
conv2d (Conv2D)             (None, 32, 32, 32)        896

conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

max_pooling2d (MaxPooling2D)  (None, 16, 16, 32)       0

max_pooling2d_1 (MaxPooling2D)   (None, 8, 8, 32)        0


dropout (Dropout)           (None, 8, 8, 32)          0

flatten (Flatten)           (None, 2048)              0

dense (Dense)               (None, 512)               1049088

dropout_1 (Dropout)         (None, 512)               0

dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 1,064,362
Trainable params: 1,064,362
Non-trainable params: 0
_________________________________________________________________
```

بعد ذلك ، نقوم بتجميع النموذج باستخدام الاعدادات التالية:

```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

تسمى دالة الخطأ التي نستخدمها فئة الانتروبيا المتقاطعة 'categorical_crossentropy'. المحسن لدينا هنا هو adam. أخيرًا ، نريد تتبع دقة نموذجنا.

حان الوقت الآن لتشغيل برنامج تدريب النموذج:

```
hist = model.fit(x_train, y_train_one_hot,
           batch_size=32, epochs=20,
           validation_split=0.2)
```

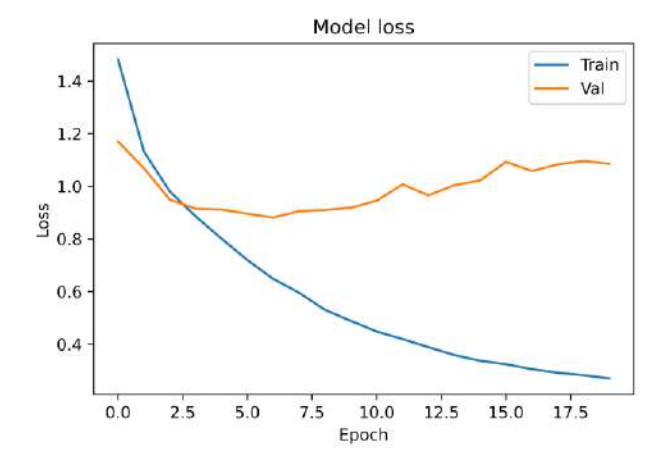
نقوم بتدريب نموذجنا بأحجام دفعات من 32 و 20 دورة. ومع ذلك ، هل لاحظت اختلافاً في الكود؟ نستخدم الإعداد validation_split = 0.2 بدلاً من validation_data. باستخدام هذا الاختصار ، لا نحتاج إلى تقسيم مجموعة البيانات الخاصة بنا إلى مجموعة تدريب ومجموعة التحقق من الصحة في البداية. بدلاً من ذلك ، نحدد ببساطة مقدار مجموعة البيانات المستخدمة كمجموعة التحقق من الصحة. في هذه الحالة ، يتم استخدام 20٪ من مجموعة البيانات الخاصة بنا كمجموعة تحقق. قم بتشغيل الخلية وسترى أن النموذج يبدأ التدريب:

```
Epoch 1/20
1250/1250 [==============================] - 27s 14ms/step - loss: 1.4824 - accuracy: 0.4623 - val_loss: 1.1698 -
val_accuracy: 0.5885
Epoch 2/20
1250/1250 [==============================] - 17s 14ms/step - loss: 1.1323 - accuracy: 0.6000 - val_loss: 1.0689 -
val_accuracy: 0.6276
Epoch 3/20
1250/1250 [==============================] - 17s 13ms/step - loss: 0.9810 - accuracy: 0.6534 - val_loss: 0.9494 -
val_accuracy: 0.6725
Epoch 4/20
1250/1250 [==============================] - 17s 14ms/step - loss: 0.8859 - accuracy: 0.6900 - val_loss: 0.9151 -
val_accuracy: 0.6828
Epoch 5/20
1250/1250 [==============================] - 18s 15ms/step - loss: 0.8015 - accuracy: 0.7171 - val_loss: 0.9117 -
val_accuracy: 0.6802
Epoch 6/20
1250/1250 [==============================] - 18s 14ms/step - loss: 0.7203 - accuracy: 0.7469 - val_loss: 0.8959 -
val_accuracy: 0.6874
Epoch 7/20
1250/1250 [==============================] - 17s 14ms/step - loss: 0.6486 - accuracy: 0.7709 - val_loss: 0.8811 -
val_accuracy: 0.7066
Epoch 8/20
1250/1250 [==============================] - 17s 14ms/step - loss: 0.5960 - accuracy: 0.7902 - val_loss: 0.9053 -
val_accuracy: 0.7055
Epoch 9/20
1250/1250 [==============================] - 17s 13ms/step - loss: 0.5312 - accuracy: 0.8129 - val_loss: 0.9100 -
val_accuracy: 0.6982
Epoch 10/20
1250/1250 [==============================] - 18s 14ms/step - loss: 0.4887 - accuracy: 0.8262 - val_loss: 0.9183 -
val_accuracy: 0.7077
Epoch 11/20
1250/1250 [==============================] - 18s 14ms/step - loss: 0.4483 - accuracy: 0.8415 - val_loss: 0.9454 -
val_accuracy: 0.7083
Epoch 12/20
1250/1250 [==============================] - 17s 13ms/step - loss: 0.4195 - accuracy: 0.8514 - val_loss: 1.0072 -
val_accuracy: 0.7062
Epoch 13/20
1250/1250 [==============================] - 18s 14ms/step - loss: 0.3889 - accuracy: 0.8625 - val_loss: 0.9655 -
val_accuracy: 0.7089
Epoch 14/20
1250/1250 [==============================] - 18s 14ms/step - loss: 0.3591 - accuracy: 0.8764 - val_loss: 1.0041 -
val_accuracy: 0.7042
Epoch 15/20
1250/1250 [==============================] - 17s 13ms/step - loss: 0.3372 - accuracy: 0.8814 - val_loss: 1.0220 -
val_accuracy: 0.7133
Epoch 16/20
1250/1250 [==============================] - 18s 14ms/step - loss: 0.3242 - accuracy: 0.8868 - val_loss: 1.0927 -
val_accuracy: 0.7041
Epoch 17/20
1250/1250 [==============================] - 23s 18ms/step - loss: 0.3053 - accuracy: 0.8925 - val_loss: 1.0579 -
val_accuracy: 0.7046
Epoch 18/20
1250/1250 [==============================] - 18s 14ms/step - loss: 0.2917 - accuracy: 0.8990 - val_loss: 1.0833 -
val_accuracy: 0.7061
Epoch 19/20
1250/1250 [==============================] - 17s 14ms/step - loss: 0.2825 - accuracy: 0.9014 - val_loss: 1.0957 -
val_accuracy: 0.7165
Epoch 20/20
```

```
1250/1250 [==============================] - 17s 14ms/step - loss: 0.2700 - accuracy: 0.9064 - val_loss: 1.0855 -
val_accuracy: 0.7131
```

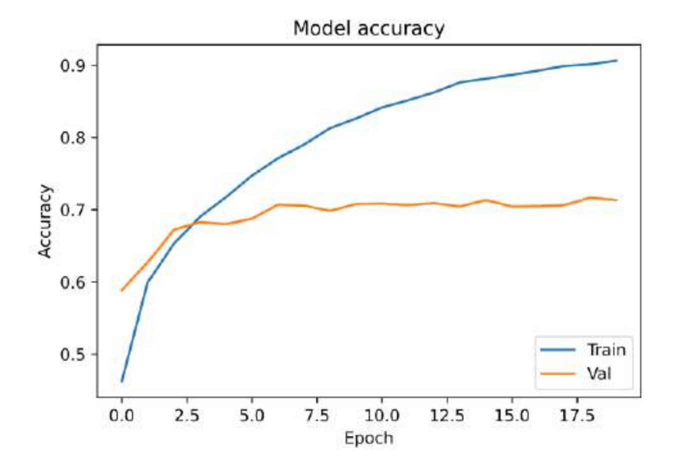
بعد الانتهاء من التدريب ، يمكننا استخدام هذا الكود ، الذي رأيناه في بناء شبكتنا العصبية الأولى، لتوضيح خطأ التدريب والتحقق من الصحة بالنسبة لعدد الدورات:

```
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```



يمكننا أيضًا توضيح الدقة:

```
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```



كما يتضح ، ان النموذج overfitting. في هذه المرحلة ، أوصي بالعودة وتجربة العديد من المعاملات مثل تغيير المعمارية أو تغيير عدد الدورات التدريبية لمعرفة ما إذا كان يمكنك الحصول على دقة val أفضل. بمجرد أن تشعر بالرضا عن النموذج الخاص بك ، يمكنك تقييمه في مجموعة الاختبار:

```
model.evaluate(x_test, y_test_one_hot)[1]
```

```
313/313 [==============================] - 2s 5ms/step - loss: 1.1410 - accuracy: 0.6924
0.6923999786376953
```

كما يتضح ، النموذج غير فعال للغاية. ومع ذلك ، فهو يعمل بشكل أفضل من التخمين العشوائي.

في هذه المرحلة ، قد ترغب في حفظ نموذجك المدرَّب (بافتراض أنك تبني نموذجًا جيد الأداء مع ضبط دقيق للمعاملات الفائقة). يتم تخزين النموذج في ملف يسمى HDF5 (مع الامتداد h5). نحفظ نموذجنا بهذا السطر من التعليمات البرمجية:

```
model.save('my_cifar10_model.h5')
```

استخدم سطر الكود التالي هذا إذاكنت تريد تنزيل النموذج المحفوظ في المستقبل:

```
from keras.models import load_model
model = load_model('my_cifar10_model.h5')
```

باختصار ، أنشأنا أول شبكة CNN لدينا لإنشاء مصنف للصور. للقيام بذلك ، استخدمنا نموذج Keras Sequential لتحديد البنية وتدريبها على مجموعات البيانات التي قمنا بمعالجتها بالفعل. لقد قمنا أيضًا بحفظ نموذجنا حتى نتمكن من استخدامه لاحقًا لتصنيف الصور دون الحاجة إلى إعادة تدريب النموذج.

الآن بعد أن أصبح لدينا نموذج ، دعنا نجربه على صورنا. للقيام بذلك ، قم بتنزيل صورة (بناءً على واحدة من فئات مجموعات البيانات العشر) من الإنترنت وضعها في نفس مجلد notebook. نستخدم صورة القطة أدناه:



ملف الصورة هو "cat.jpg". نقرأ الآن ملف JPEG هذا كمجموعة من قيم البكسل:

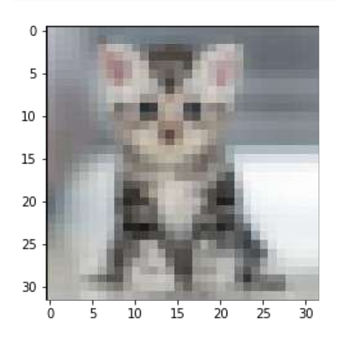
```
my_image = plt.imread("cat.jpg")
```

أول شيء يتعين علينا القيام به هو تغيير حجم الصورة القطة لدينا حتى نتمكن من وضعها في نموذجنا (حجم الإدخال 3 × 32 × 32). بدلاً من برمجة دالة تغيير الحجم بأنفسنا ، دعنا نستخدم حزمة تسمى "scikit-image" لمساعدتنا في القيام بذلك:

```
from skimage.transform import resize
my_image_resized = resize(my_image, (32,32,3))
```

يمكننا رؤية صورتنا التي تم تغيير حجمها على النحو التالي:

```
img = plt.imshow(my_image_resized)
```



لاحظ أن حجم الصورة التي تم تغيير حجمها يحتوي على قيم بكسل تم قياسها بالفعل بين 0 و 1 ، لذلك لا نحتاج إلى تطبيق خطوات المعالجة المسبقة التي قمنا بها سابقًا لصورة البرنامج التدريبي الخاصة بنا. الآن ، باستخدام الكود `model.predict` ، نرى كيف سيبدو ناتج نموذجنا المدرب عند إعطائنا صورة قطة:

```
import numpy as np
probabilities = model.predict(np.array( [my_image_resized,] ))
```

ناتج الكود أعلاه هو ناتج 10 خلايا عصبية مرتبطة بتوزيع الاحتمالات على الفئات. إذا قمنا بتشغيل الخلية ، سيكون لدينا:

```
probabilities
```

```
array([[2.6720341e-02, 3.1344647e-05, 1.5095539e-01, 3.8518414e-01,
        3.3354717e-03, 3.2324010e-01, 5.1648129e-02, 5.7933435e-02,
        9.2716294e-04, 2.4454062e-05]], dtype=float32)
```

قم بتنفيذ مقتطف الكود التالي لقراءة توقعات النموذج بسهولة:

```
number_to_class = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
index = np.argsort(probabilities[0,:])
print("Most likely class:", number_to_class[index[9]], "-- Probability:", probabilities[0,index[9]])
print("Second most likely class:", number_to_class[index[8]], "-- Probability:", probabilities[0,index[8]])
print("Third most likely class:", number_to_class[index[7]], "-- Probability:", probabilities[0,index[7]])
print("Fourth most likely class:", number_to_class[index[6]], "-- Probability:", probabilities[0,index[6]])
print("Fifth most likely class:", number_to_class[index[5]], "-- Probability:", probabilities[0,index[5]])
```

```
Most likely class: cat -- Probability: 0.31140402
Second most likely class: horse -- Probability: 0.296455
Third most likely class: dog -- Probability: 0.1401798
Fourth most likely class: truck -- Probability: 0.12088975
Fifth most likely class: frog -- Probability: 0.078746535
```

كما ترى ، توقع النموذج بدقة أن الصورة المدخلة كانت في الواقع صورة قطة. ومع ذلك ، هذا ليس أفضل نموذج لدينا ودقته منخفضة جداً ، لذلك لا يجب أن تتوقع الكثير منه. يغطي هذا المثال أساسيات شبكات CNN فقط في مجموعة بيانات بسيطة جداً. كتمرين ، يمكنك بناء نماذج أخرى لمجموعة البيانات هذه ومقارنة النتائج.

خلاصة الفصل

- **تستخدم الشبكات العصبية الالتفافية الالتفاف بدلاً من مضاعفة المصفوفة في طبقة واحدة على الأقل من طبقاتها.**
- **يؤدي تحديد الميزات من خلال تطبيق الفلاتر إلى إنتاج خريطة الميزات.**
- **يشار إلى طبقة الالتفاف أيضًا باسم طبقة استخراج الميزات. لأنه يتم استخراج خصائص الصورة في هذه الطبقة.**

اختبار

1. **ما هي المعمارية العامة للشبكة الالتفافية؟**
2. **ما هي السمات المميزة الثلاثة للشبكات الالتفافية؟**
3. **ما هي أجزاء طبقة الالتفاف؟**
4. **ما هي فوائد استخدام الالتفاف؟**
5. **ما هو دور طبقة التجميع في الشبكات الالتفافية؟**

5 الشبكة العصبية المتكررة

اهداف التعليم

- ما هي الشبكة العصبية المتكررة؟
- التعرف على LSTM.
- توليد النص وتصنيفه مع الشبكات العصبية المتكررة.

المقدمة

تتلقى بنية الشبكة العصبية التي تمت مناقشتهافي الفصول السابقة مدخلات ذات حجم ثابت وتوفر مخرجات ذات حجم ثابت. يقدم لنا هذا الفصل مقدمة عن الشبكات العصبية المتكررة **(Recurrent Neural Networks)**، أو اختصارا RNN. تساعدنا شبكات RNNفي التعامل مع التسلسلات متغيرة الطول من خلال تحديد حلقة التغذية الراجعة على هذه التسلسلات**(sequences)**. تجعل القدرة على معالجة تسلسل الإدخال المخصص شبكات RNN قابلة للاستخدامفي مهام مثل نمذجة اللغة والتعرف على الكلام والمزيد.في الواقع، من الناحية النظرية، يمكن تطبيق RNNs على أي مشكلة لأنه ثبت أنها **Turing-Complete**[1]. هذا يعني أنه من الناحية النظرية، يمكنهم محاكاة أي برنامج لا يستطيع الكمبيوتر العادي حسابه. على سبيل المثال، اقترح Google DeepMind نموذجًا يسمى آلات تورينج العصبية التي يمكن أن تعلمك كيفية تشغيل خوارزميات بسيطة مثل الفرز.

ما هي الشبكة العصبية المتكررة؟

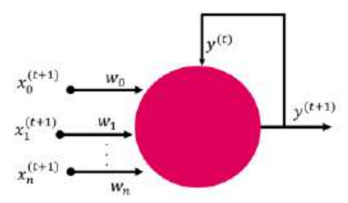
في الفصل السابق، وصفنا بنية الشبكات العصبية الالتفافية التي تشكل الأساس للعديد من أنظمة الرؤية الحاسوبية المتقدمة. ومع ذلك، فإننا لا ندرك العالم من حولنا بالبصر وحده. على سبيل المثال، يلعب الصوت أيضًا دورًا مهمًا جدًا. وبشكل أكثر تحديدًا، نحب نحن البشر التواصل والتعبير عن الأفكار والأفكار المعقدة من خلال تسلسلات رمزية وتمثيلات مجردة. ومن ثم، فمن المنطقي أننا نريد أن تكون الآلات قادرة على فهم المعلومات المتسلسلة.

عندما يتم ترتيب البيانات بحيث يكون لكل قطعة نوع من العلاقة مع القطع التي تم إنشاؤها قبلها وبعدها، يشار إليها باسم التسلسلات. **الشبكات العصبية المتكررة** هي نوع من الشبكات العصبية الاصطناعية المصممة لاكتشاف الأنماطفي البيانات المتسلسلة مثل النص المادة الوراثية والكتابة اليدوية والكلمات المنطوقة وبيانات السلاسل الزمنية وأسواق الأسهم وما إلى ذلك. الفكرة من وراء هذه الشبكات العصبية هي أنها تسمح للخلايا بالتعلم من الخلايا المرتبطة سابقًا. يمكن القول إن هذه الخلايا لها "ذاكرة" بطريقة ما. ومن ثم، فإنهم يبنون معرفة أكثر تعقيدًا من بيانات الإدخال.

تشترك النماذج التي تمت دراستها في الفصول السابقة في شيء واحد. بعد الانتهاء من عملية التدريب ، يتم إصلاح الأوزان وتعتمد المخرجات على عينة الإدخال فقط. من الواضح أن هذا السلوك متوقع من المصنف ، ولكن هناك العديد من السيناريوهات التي يجب على المتنبئ فيها مراعاة محفوظات قيم الإدخال. السلسلة الزمنية هي مثال كلاسيكي على ذلك. لنفترض أن علينا

[1] Alex Graves and Greg Wayne and Ivo Danihelka (2014). "Neural Turing Machines".

توقع درجة حرارة الأسبوع المقبل. إذا حاولنا استخدام آخر قيمة معروفة لـ $x(t)$ و MLP مدرب للتنبؤ بـ $x(t+1)$، فلا يمكننا النظر في الظروف الزمنية مثل الموسم وتاريخ الموسم على مر السنين وما إلى ذلك. سيكون الانحدار قادرًا على ربط المخرجات التي تنتج أدنى متوسط للخطأ، ولكن في المواقف الحقيقية ، هذا لا يكفي. الطريقة المعقولة الوحيدة لحل هذه المشكلة هي تحديد بنية جديدة للخلايا العصبية الاصطناعية لتوفير ذاكرة لها. يظهر هذا المفهوم في الشكل أدناه:



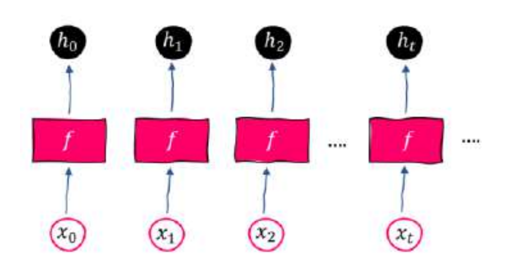
لم تعد الخلية العصبية وحدة حوسبة امامية التغذية تمامًا ، حيث أن اتصالها الراجع يجبرها على تذكر ماضيها واستخدامها للتنبؤ بقيم جديدة.

تقضي الشبكات العصبية المتكررة على أوجه القصور في الشبكات العصبية أمامية التغذية، وذلك لأن الشبكات العصبية أمامية التغذية يمكنها فقط قبول مدخلات ذات حجم ثابت وتنتج فقط مخرجات ذات حجم ثابت وغير قادرة على النظر في المدخلات السابقة بنفس الترتيب. من خلال النظر في المدخلات السابقة في التسلسل، تكون الشبكة العصبية المتكررة قادرة على التقاط التبعيات عندما تكون الشبكة العصبية امامية التغذية غير قادرة على ذلك.

تأخذ الشبكات العصبية المتكررة تسلسلاً كمدخلات وتقيم الشبكة العصبية لكل خطوة زمنية. يمكن اعتبار هذه الشبكات على أنها شبكة عصبية لها حلقة تسمح لها بالحفاظ على الحالة. عند التقييم، تفتح الحلقة من خلال الخطوات الزمنية للتسلسل. هذه الحلقات أو الروابط المتكررة هي سبب تسمية هذه الشبكات بشبكات المتكررة. حقيقة أن الشبكة العصبية امامية التغذية تتكون من حلقة تعني أنه يمكن إرجاع ناتج خلية عصبية واحدة في نقطة زمنية واحدة إلى نفس الخلية العصبية في نقطة زمنية أخرى. والنتيجة هي أن الشبكة لديها ذاكرة لعمليات التنشيط السابقة (وبالتالي المدخلات السابقة التي لعبت دورًا في هذا التنشيط).

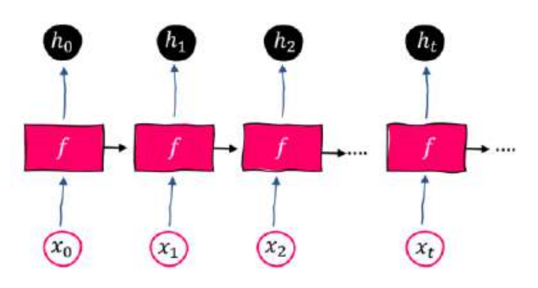
هيكل الشبكة العصبية المتكررة

افترض شبكة عصبية تقليدية ، كما هو موضح في الشكل أدناه:



لدينا عدد من المدخلات x_t ، حيث تمثل t خطوة زمنية أو تسلسل. لنفترض أن مدخلات t المختلفة مستقلة عن بعضها البعض. يمكن كتابة ناتج الشبكة في كل t كـ $h_t = f(x_t)$.

في RNNs ، تنقل حلقة التغذية الراجعة معلومات الحالة الحالية إلى الحالة التالية ، كما هو موضح في الإصدار المفتوح من الشبكة ، كما هو موضح أدناه.:



يمكن كتابة ناتج شبكة RNN في كل t كـ $h_t = f(h_{t-1},\ x_t)$. يتم إجراء نفس الشيء f على كل عنصر من عناصر التسلسل ، ويعتمد ناتج h_t على ناتج الحسابات السابقة. وبالتالي ، على عكس الشبكات التقليدية ، حيث تعتمد الحالة فقط على المدخلات الحالية (ووزن الشبكة) ، هنا h_t هي دالة للإدخال الحالي بالإضافة إلى الحالة السابقة h_{t-1}. يمكنك اعتبار h_{t-1} بمثابة ملخص لجميع مدخلات الشبكة السابقة.

بفضل بُنية السلسلة هذه ، أو بعبارة أخرى ، التخزين الإضافي (الذاكرة) مما تم حسابه حتى الآن ، تم تحقيق نجاح كبير في استخدام RNN في السلاسل الزمنية والبيانات المتسلسلة.

تحصل RNNs على اسمها من هذا لأنها تطبق نفس الدالة بشكل متكرر على التسلسلات.

تحتوي RNN على ثلاث مجموعات من المعاملات (الوزن):

- U يحول الإدخال x_t إلى h_t .
- W يحول الحالة السابقة h_{t-1} إلى الحالة الحالية h_t
- V يعيّن الحالة الداخلية المحسوبة حديثًا لـ h_t إلى الناتج y .

تطبق U و W و V التحويل الخطي على المدخلات المقابلة. الحالة الأساسية لمثل هذا التحويل هي مجموع اوزان مانعرفه. يمكننا الآن تحديد الحالة الداخلية ومخرجات الشبكة على النحو التالي:

$$h_t = f(h_{t-1} * W + x_t * U)$$

$$y_t = h_t * V$$

هنا ، f هي دالة تنشيط غير خطية.

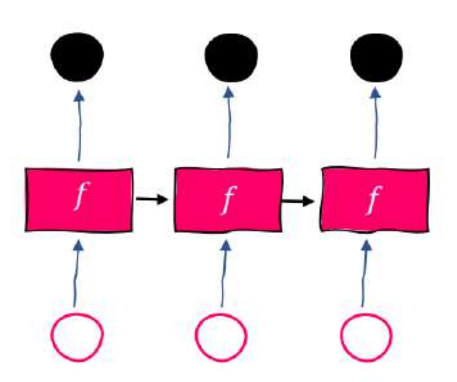
لاحظ أنه في RNN ، تعتمد كل حالة على جميع الحسابات السابقة من خلال هذه العلاقة التكرارية. من الدلالات المهمة أن RNNs لها ذاكرة بمرور الوقت ، لأن حالات h تحتوي على معلومات تستند إلى الخطوات السابقة. من الناحية النظرية ، يمكن لـ RNNs تخزين المعلومات طالما رغبوا في ذلك. ، لكن في الممارسة العملية يمكنهم فقط النظر إلى الوراء بضع خطوات.

في RNN، تعتمد كل حالة على جميع الحسابات السابقة بواسطة المعادلة التكرارية. ومن النتائج المهمة لذلك إنشاء الذاكرة بمرور الوقت، لأن الحالات تستند إلى مراحل سابقة.

انواع معماريات RNN

نظرًا لأن RNNs لا تقتصر على معالجة المدخلات ذات الحجم الثابت ، فهي تتضمن بنى مختلفة:

- **واحد لواحد:** كما يمكن رؤيته في الشكل المقابل ، في هذه البُنية ، يتم تعيين وحدة إدخال RNN إلى وحدة مخفية ووحدة إخراج. هذه البُنية هي عملية متسلسلة مثل الشبكات العصبية أمامية التغذية والشبكات العصبية الالتفافية. مثال على هذه العملية هو تصنيف الصور.
- **واحد إلى متعدد:** كما يتضح من الشكل أدناه ، في هذه البُنية ، يتم تعيين وحدة إدخال RNN إلى عدة وحدات مخفية وعدة وحدات إخراج. المثال العملي لهذه البُنية هو وصف الصور. تتلقى طبقة الإدخال صورة وتعيينها إلى عدة كلمات.
- **متعدد إلى واحد:** كما يتضح من الشكل المقابل ، في هذه البنية ، يتم تعيين عدة وحدات إدخال RNN إلى عدة وحدات مخفية ووحدة إخراج واحدة. مثال عملي على هذه البُنية هو تصنيف المشاعر (Sentiment analysis). تتلقى طبقة الإدخال إشارات متعددة من كلمات الجملة وترسمها على أنها عاطفة إيجابية أو سلبية.



- **متعدد لمتعدد:** كما يتضح من الشكل المقابل ، في هذه البُنية ، يتم تعيين العديد من وحدات إدخال RNN إلى عدة وحدات مخفية وعدة وحدات إخراج. مثال عملي على هذه البُنية هو الترجمة الآلية. تستقبل طبقة الإدخال عدة أحرف من كلمات اللغة المصدر وتربطها بأحرف الكلمات في اللغة الهدف.

توليد نص باستخدام RNN

تُستخدم RNNs بشكل شائع كنماذج لغوية (language models)في مجال معالجة اللغة الطبيعية (natural language processing). نعتزم العمل على لغة مثيرة للاهتمام نسبيًا لنمذجة توليد النص، حيث تُستخدم نماذج RNN لتعلم التسلسلات النصية لمجال معين ثم لإنشاء تسلسل نصي جديد تمامًا ومعقول في المجال. يمكن لمولِّد النص المستند إلى RNN أن يأخذ أي نص إدخال، كالروايات مثل هاري بوتر، وأشعار شكسبير وسيناريوهات لأفلام مثل حرب النجوم، وإنتاج قصائد شكسبير وسيناريوهات حرب النجوم. إذا كان النموذج مدربًا جيداً، فيجب أن يكون التركيب مقبولًا وقراءته مشابهة للنص الأصلي.في هذا الجزء نستخدم رواية "الحرب والسلام" للمؤلف الروسي ليو تولستوي كمثال. ومع ذلك، يمكنك استخدام أي من كتبك المفضلة كمدخلات تدريبية. يعد مشروع جوتنبرج (www.gutenberg.org) مصدرًا رائعًا لذلك، حيث نفد طباعة ما يزيد عن 57000 كتاب رائع مجانًا.

أولا عليك تحميل ملف warpeace_input.txt مباشرة من الرابط:

https://cs.stanford.edu/people/karpathy/char-rnn/warpeace_input.txt

من ناحية أخرى، يمكننا تنزيله من مشروع جوتنبرج.

https://www.gutenberg.org/ebooks/2600

نحمل الملف، لكننا بحاجة إلى إجراء بعض التنظيف. ثم نقرأ الملف ونحول النص إلى أحرف صغيرة ونلقي نظرة سريعة عليه بطباعة أول 100 حرف:.

```
training_file = 'warpeace_input.txt'
raw_text = open(training_file, 'r').read()
raw_text = raw_text.lower()
raw_text[:100]
```

```
'ufeff"well, prince, so genoa and lucca are now just family estates
of thenbuonapartes. but i warn you, i'
```

الآن نحن بحاجة إلى حساب عدد الأحرف:

```
n_chars = len(raw_text)
```

```
print('Total characters: {}'.format(n_chars))
```

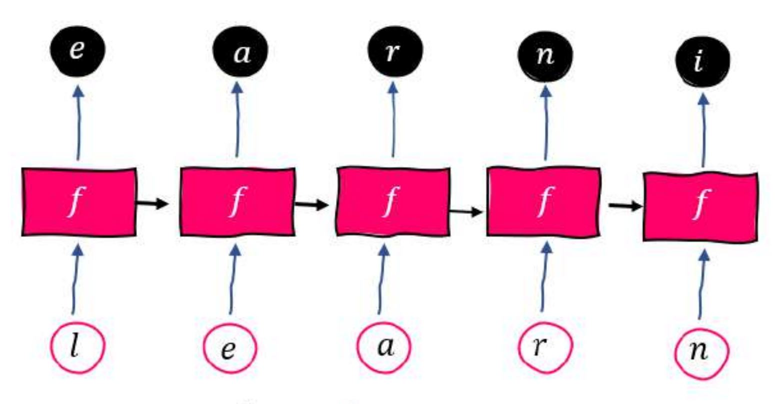
```
Total characters: 3196213
```

بعد ذلك، يمكننا الحصول على أحرف وأحجام كلمات فريدة:

```
chars = sorted(list(set(raw_text)))
n_vocab = len(chars)
print('Total vocabulary (unique characters): {}'.format(n_vocab))
print(chars)
```

```
Total vocabulary (unique characters): 57
['\n', ' ', '!', '"', "'", '(', ')', '*', ',', '-', '.', '/', '0',
'1', '2', '3', '4', '5', '6', '7', '8', '9', ':', ';', '=', '?',
'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',
'à', 'ä', 'é', 'ê', '\ufeff']
```

الآن، لدينا مجموعة بيانات تدريبية خام تتكون من أكثر من 3 ملايين حرف و57 حرفًا فريدًا. ولكن كيف يمكننا تغذيته لنموذج RNN؟في معمارية متعدد لمتعدد، يأخذ النموذج المتتاليات وينتج المتتالياتفي وقت واحد.في حالتنا، يمكننا تغذية النموذج بتسلسلات أحرف ثابتة الطول. طول متتاليات الإخراج يساوي متتاليات الإدخال، ويتم نقل حرف واحد من تسلسلات الإدخال الخاصة بهم. لنفترض أننا قمنا بتعيين طول التسلسل على 5 لكلمة learning. يمكننا الآن إنشاء عينة تدريبية مع مدخلات learn ومخرجات earni. يمكننا توضيح هذا الإجراءفي الشبكة على النحو التالي:



لقد قمنا للتو بعمل تدريبي. بالنسبة لمجموعة البرامج التدريبية بأكملها، يمكننا تقسيم بيانات النص الخام إلى تسلسلات متساوية الطول، على سبيل المثال 100. كل تسلسل من أحرف الإدخال هو مثال تدريب. ثم نقوم بتحليل بيانات النص الخام بنفس الطريقة، ولكن هذه المرة نبدأ بالحرف الثاني. كل تسلسل ناتج تم الحصول عليه هو عينة تدريب. على سبيل المثال، بالنظر

إلى النص الخام لـ deep learning architectures والرقم 5 بطول التسلسل، يمكننا بناء خمسة أمثلة تدريبية على النحو التالي:

المدخلات	المخرجات
deep_	eep_l
learn	earni
ing_a	ng_ar
rchit	chite
ectur	cture

هنا، يشير _ إلى المساحة الفارغة.

نظرًا لأن نماذج الشبكة العصبية لا تتلقى سوى البيانات الرقمية، يتم تمثيل تسلسل الإدخال والإخراج للأحرف بواسطة متجهات ترميز one-hot. نقوم بإنشاء قاموس عن طريق تعيين 57 حرفًا للمؤشرات من 0 إلى 56:

```
index_to_char = dict((i, c) for i, c in enumerate(chars))
char_to_index = dict((c, i) for i, c in enumerate(chars))
print(char_to_index)
```

على سبيل المثال، يتم تحويل الحرف *e* إلى متجه بطول 57 بقيمة 1 في الفهرس 30 وجميع قيمه الأخرى هي 0 (لقد رأيت كيف تم تشفير هذا في الفصل السابق). بمجرد أن يصبح جدول البحث عن الاحرف جاهزًا، يمكننا إنشاء مجموعة بيانات التدريب على النحو التالي:

```
import numpy as np
seq_length = 100
n_seq = int(n_chars / seq_length)
```

من خلال ضبط طول التسلسل على 100، سيكون لدينا عينة n_seq. بعد ذلك، نقوم بتهيئة مدخلات ومخرجات التدريب:

```
X = np.zeros((n_seq, seq_length, n_vocab))
Y = np.zeros((n_seq, seq_length, n_vocab))
```

لاحظ أن طول التسلسل كما يلي:

(رقم العينة، طول التسلسل، أبعاد الميزة)

مثل هذا النموذج مطلوب، لأننا نعتزم استخدام Keras لتعليم نموذج RNN. بعد ذلك، تقوم بإنشاء كل مثيل n_seq:

```
for i in range(n_seq):
        x_sequence = raw_text[i * seq_length : (i + 1) * seq_length]
        x_sequence_ohe = np.zeros((seq_length, n_vocab))
        for j in range(seq_length):
                char = x_sequence[j]
                index = char_to_index[char]
                x_sequence_ohe[j][index] = 1.
        X[i] = x_sequence_ohe
```

```
        y_sequence = raw_text[i * seq_length + 1 : (i + 1) * seq_length + 1]
        y_sequence_ohe = np.zeros((seq_length, n_vocab))
        for j in range(seq_length):
                char = y_sequence[j]
                index = char_to_index[char]
                y_sequence_ohe[j][index] = 1.
        Y[i] = y_sequence_ohe
```

إذا كنت ترغب في ذلك، يمكنك رؤية المصفوفة بتنفيذ الخلايا أدناه:

```
X.shape
```

(31962, 100, 57)

```
Y.shape
```

(31962, 100, 57)

حتى الآن، قمنا بإعداد مجموعة بيانات التدريب وحان الوقت الآن لبناء نموذج RNN. أدخل أولاً جميع الوحدات الضرورية:

```
from keras.models import Sequential
from keras.layers.core import Dense, Activation, Dropout
from keras.layers.recurrent import SimpleRNN
from keras.layers.wrappers import TimeDistributed
from keras import optimizers
from tensorflow import keras
```

الآن، نحدد المعاملات الفائقة، بمافي ذلك حجم الدفعة، وعدد الخلايا العصبية، وعدد الطبقات، واحتمال الحذف العشوائي، وعدد الدورات:

```
batch_size = 100
n_layer = 2
hidden_units = 800
n_epoch = 300
dropout = 0.3
```

ثم نقوم بإنشاء الشبكة:

```
model = Sequential()
model.add(SimpleRNN(hidden_units, activation='relu', input_shape=(None,
n_vocab), return_sequences=True))
model.add(Dropout(dropout))
for i in range(n_layer - 1):
    model.add(SimpleRNN(hidden_units, activation='relu',
return_sequences=True))
    model.add(Dropout(dropout))
model.add(TimeDistributed(Dense(n_vocab)))
model.add(Activation('softmax'))
```

هناك بعض الأشياء التي يجب وضعهافي الاعتبار بشأن النموذج الذي أنشأناه:

- return_sequences=True: يصبح ناتج طبقات العودة تسلسلاً ، بُنية كمتعدد لمتعدد تجعل ذلك ممكناً ، كما أردنا أن يكون. خلاف ذلك، يصبح متعدد إلى واحد وسيكون العنصر الأخير هو الناتج.

- TimeDistributed: نظرًا لأن إخراج طبقات العودة عبارة عن تسلسل ، فإن الطبقة التالية هي طبقة متصلة كاملة ولا تتلقى مدخلات متتالية. يتم استخدام TimeDistributed لتجاوز هذا.

بالنسبة للمحسن، اخترنا RMSprop بمعدل تعليم 0.001:

```
optimizer = keras.optimizers.RMSprop(learning_rate=0.001, rho=0.9,
epsilon=1e-08, decay=0.0)
```

بإضافة خطأ إنتروبيا متعددة الطبقات "categorical_crossentropy" ، ننتهي من بناء نموذجنا وأخيراً نقوم بتجميع النموذج:

```
model.compile(loss= "categorical_crossentropy", optimizer=optimizer)
```

باستخدام الكود التالي، يمكننا إلقاء نظرة على ملخص النموذج:

```
model.summary()
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn_4 (SimpleRNN)    (None, None, 800)         686400

 dropout_4 (Dropout)         (None, None, 800)         0

 simple_rnn_5 (SimpleRNN)    (None, None, 800)         1280800

 dropout_5 (Dropout)         (None, None, 800)         0

 time_distributed_2 (TimeDis  (None, None, 57)         45657
 tributed)

 activation_2 (Activation)   (None, None, 57)          0

=================================================================
Total params: 2,012,857
Trainable params: 2,012,857
Non-trainable params: 0
```

لدينا أكثر من 2 مليون معامل للتدريب. ولكن قبل بدء عملية التدريب الطويلة، من الممارسات الجيدة إعداد بعض عمليات الاسترجاع callback لتتبع الإحصائيات والحالة الداخلية للنموذج أثناء التدريب. تشمل دوال callback ما يلي:

- **نقطة التحقق (checkpoint) النموذج** ، والتي تخزن النموذج بعد كل فترة حتى نتمكن من تحميل أحدث نموذج محفوظ واستئناف التدريب من هناك في حالة التوقف غير المتوقع.
- **التوقف المبكر** ، والذي يوقف التدريب عندما لا تتحسن دالة الخطأ.
- **مراجعة نتائج توليد النص بشكل منتظم**. نريد أن نرى مدى معقولية النص الناتج لأن عيوب التدريب ليست ملموسة بما فيه الكفاية.

يتم تعريف أو تهيئة هذه الدوال على النحو التالي:

```
from keras.callbacks import Callback, ModelCheckpoint, EarlyStopping

filepath="weights/weights_layer_%d_hidden_%d_epoch_{epoch:03d}_loss_{loss
:.4f}.hdf5"
```

```
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,
save_best_only=True, mode='min')
```

يتم تخزين نقاط التحقق النموذجيةفي اسم الملف مع رقم الدورة التدريبية وخطأ التدريب. نراقب أيضًا اخطاء الاعتماديفي نفس الوقت لمعرفة ما إذا كان تقليل الاخطاء سيتوقف لمدة 50 فترة متتالية:

```
early_stop = EarlyStopping(monitor='loss', min_delta=0, patience=50,
verbose=1, mode='min')
```

بعد ذلك، نحتاج إلى callback لمراقبة الجودة. نكتب أولاً دالة مساعدة تولد نصًا بأي طول وفقًا لنموذج RNN الخاص بنا:

```
def generate_text(model, gen_length, n_vocab, index_to_char):
    """
    Generating text using the RNN model
    @param model: current RNN model
    @param gen_length: number of characters we want to generate
    @param n_vocab: number of unique characters
    @param index_to_char: index to character mapping
    @return:
    """
    # Start with a randomly picked character
    index = np.random.randint(n_vocab)
    y_char = [index_to_char[index]]
    X = np.zeros((1, gen_length, n_vocab))
    for i in range(gen_length):
        X[0, i, index] = 1.
        indices = np.argmax(model.predict(X[:, max(0, i - 99):i + 1, :])[0], 1)
        index = indices[-1]
        y_char.append(index_to_char[index])
    return ('').join(y_char)
```

تبدأ هذه الدالة بحرف تم اختياره عشوائيًا. يتنبأ نموذج الإدخال بعد ذلك بكل حرف من أحرف gen_length-1 المتبقية بناءً على الأحرف التي تم إنشاؤها مسبقًا والتي يصل طولها إلى 100 (طول التسلسل). يمكننا الآن تحديد فئة callback تقوم بإنشاء نص لكل فترة N:

```
class ResultChecker(Callback):
    def __init__(self, model, N, gen_length):
        self.model = model
        self.N = N
        self.gen_length = gen_length

    def on_epoch_end(self, epoch, logs={}):
        if epoch % self.N == 0:
            result = generate_text(self.model, self.gen_length, n_vocab,
index_to_char)
            print('\nMy War and Peace:\n' + result)
```

الآن بعد أن أصبحت جميع المكونات جاهزة، نبدأ بتدريب النموذج:

```
model.fit(X, Y, batch_size=batch_size, verbose=1, epochs=n_epoch,
                 callbacks=[ResultChecker(model, 10, 200), checkpoint, early_stop])
```

يكتب المولد 200 حرفًا لكل 10 فترات. النتائج التالية خاصة بالفترات 1 و11 و51 و101:

Epoch 1:

```
Epoch 1/300
8000/31962 [======>.......................] - ETA: 51s - loss: 2.8891
31962/31962 [==============================] - 67s 2ms/step - loss: 2.1955
My War and Peace:
5 the count of the stord and the stord and the stord and the stord and the
stord and the stord and the stord and the stord and the stord and the stord
and the and the stord and the stord and the stord
Epoch 00001: loss improved from inf to 2.19552, saving model to
weights/weights_epoch_001_loss_2.19552.hdf5
```

Epoch 11:

```
Epoch 11/300
100/31962 [..............................] - ETA: 1:26 - loss: 1.2321
31962/31962 [==============================] - 66s 2ms/step - loss: 1.2493
My War and Peace:
?" said the countess was a strange the same time the countess was already
been and said that he was so strange to the countess was already been and
the same time the countess was already been and said
Epoch 00011: loss improved from 1.26144 to 1.24933, saving model to
weights/weights_epoch_011_loss_1.2493.hdf5
```

Epoch 51:

```
Epoch 51/300
31962/31962 [==============================] - 66s 2ms/step - loss: 1.1562
My War and Peace:
!!CDP!E.agrave!! to see him and the same thing is the same thing to him and
the same thing the same thing is the same thing to him and the sam thing
the same thing is the same thing to him and the same thing the sam
Epoch 00051: loss did not improve from 1.14279
```

Epoch 101:

```
Epoch 101/300
31962/31962 [==============================] - 67s 2ms/step - loss: 1.1736
My War and Peace:
= the same thing is to be a soldier in the same way to the soldiers and the
same thing is the same to me to see him and the same thing is the same to
me to see him and the same thing is the same to me
Epoch 00101: loss did not improve from 1.11891
```

توقف التدريب في وقت مبكر من الدورة 203:

Epoch 00203: loss did not improve from 1.10864
Epoch 00203: early stopping

يولد النموذج النص التالي في الفترة 151:

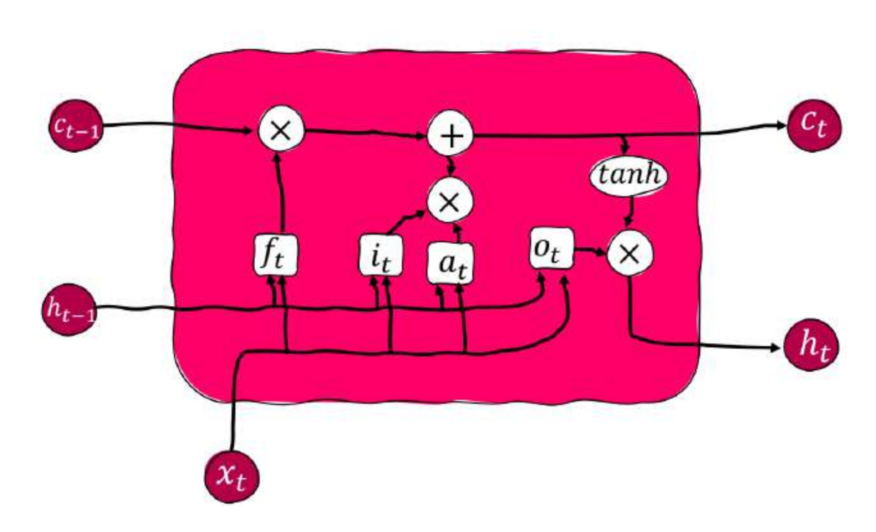
```
which was a strange and serious expression of his face and shouting and said
that the countess was standing beside him. "what a battle is a strange and
serious and strange and so that the countess was
```

تُقرأ "الحرب والسلام" الخاصة بنا جيدًا إلى حد ما. ومع ذلك، قد تتساءل عما إذا كان بإمكاننا القيام بعمل أفضل من خلال تغيير المعاملاتفي نموذج RNN هذا؟ الجواب نعم، لكن الأمر لا يستحق ذلك. لأن تدريب نموذج RNN لحل المشكلات التي تتطلب تعلم الاعتماد على المدى الطويل ليس فعالًا للغاية. تم تصميم الهياكل مثل LSTM وGRU خصيصًا لحل هذه المشكلة.

LSTM

من الناحية النظرية، يمكن لـ RNNs البسيطة تعلم التبعيات طويلة المدى، ولكنفي الممارسة العملية، بسبب مشكلة تلاشي التدرج، فإنها تقتصر على تعلم التبعيات قصيرة المدى. لمواجهة هذا القيد، تم تقديم شبكة الذاكرة طويلة المدى **(Long short term memory)** او **(LSTM)**. يمكن لـ LSTM تكوين تبعيات طويلة المدى بسبب وجود خلية ذاكرة خاصةفي هيكلها.

الفكرة الأساسية لـ LSTM هي خلية الحالة **(cell state)** التي يمكن فيها كتابة المعلومات أو حذفها بشكل صريح. تظهر خلية الحالة هذه للوقت t كـ c_t في الشكل 5ـ1.



$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$
$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$
$$a_t = \tanh(W_c h_{t-1} + U_c x_t + b_f)$$
$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$
$$c_t = f_t * c_{t-1} + i_t * a_t$$
$$h_t = o_t * \tanh\,(c_t)$$

شكل 5-1. معمارية LSTM

لا يمكن تغيير خلية حالة LSTM إلا عن طريق بوابات معينة يتم من خلالها نقل المعلومات. يتكون LSTM النموذجي من ثلاث بوابات: **بوابة النسيان (f) وبوابة الإدخال (i) وبوابة الإخراج (o)**.

البوابة الأولى في LSTM هي **بوابة النسيان**. تقرر هذه البوابة ما إذا كنا نريد مسح خلية الحالة أم لا. يعتمد قرار البوابة المنسية على المخرجات السابقة h_{t-1} والمدخلات الحالية x_t. تُستخدم sigmoid لتوليد إخراج بين صفر وواحد لكل عنصر في خلية الحالة. يتم تنفيذ الضرب بين مخرج بوابة النسيان وخلية الحالة. تعني قيمة واحد في مخرج بوابة النسيان أن معلومات العنصر مخزنة بالكامل في خلية الحالة. في المقابل، الصفر يعني النسيان الكامل للمعلومات في عنصر خلية الحالة. هذا يعني أن LSTM يمكنها إزالة المعلومات غير ذات الصلة من متجه خلية الحالة الخاص بها. معادلة بوابة النسيان كالتالي:

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$

تقرر البوابة التالية (**بوابة الإدخال**) إضافة معلومات جديدة إلى خلية الذاكرة. يتم ذلك في جزأين: **تحديد القيم المراد تحديثها ثم إنشاء القيم المراد تحديثها**. يتم استخدام المتجه i_t أولاً لتحديد قيم المرشحين الجدد المحتملين لتضمينها في خلية الحالة. يحتوي متجه المرشح a_t أيضًا على مصفوفة وزن خاصة به ويستخدم الحالة والمدخلات المخفية السابقة لتشكيل متجه بأبعاد مماثلة لخلية الحالة. لإنشاء هذا المتجه المرشح ، يتم استخدام دالة tanh كدالة غير خطية. تظهر هذه العملية في المعادلات التالية:

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$
$$a_t = \tanh(W_c h_{t-1} + U_c x_t + b_f)$$

تحدد بوابة النسيان كيفية تحديث خلية الحالة في كل خطوة زمنية. يتم تحديث خلية الحالة في خطوة زمنية عبر المعادلة التالية:

$$c_t = f_t * c_{t-1} + i_t * a_t$$

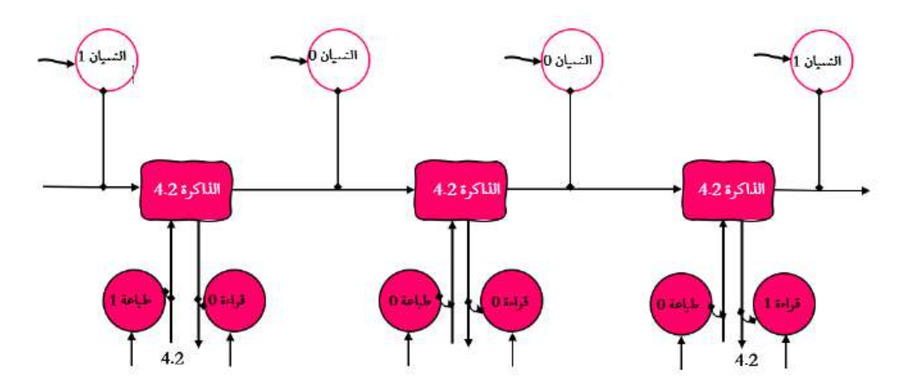
البوابة الأخيرة (**بوابة الإخراج**) تقرر ما هو الإخراج. الناتج النهائي لخلية LSTM هو الحالة المخفية h_t. بوابة الإخراج تأخذ h_{t-1} و x_t كمدخل. أولاً، تُستخدم sigmoid لحساب المتجه بقيم بين صفر وواحد لتحديد قيم خلية الحالة في الخطوة الزمنية. ثم قمنا بتعيين قيمة خلية الحالة إلى طبقة tanh لمضاعفة قيمتها أخيرًا بمخرجات الطبقة السابقة sigmoid، بحيث تتم مشاركة الأجزاء المرغوبة في الإخراج. يعني الإخراج 0 أن كتلة الخلية لا تنتج أي معلومات، بينما يعني الإخراج 1 أن الذاكرة الكاملة لكتلة الخلية يتم نقلها إلى إخراج الخلية. تظهر المعادلات التالية هذا الاتجاه:

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$

$$h_t = o_t * \tanh (c_t)$$

الآن كيف تحمينا LSTM من تلاشي التدرج؟ لاحظ أنه إذا كانت بوابة النسيان 1 وكانت بوابة الإدخال 0، فسيتم نسخ حالة الخلية بالضبط خطوة بخطوة. فقط بوابة النسيان يمكنها محو ذاكرة الخلية تمامًا. نتيجة لذلك، يمكن أن تظل الذاكرة دون تغيير لفترة طويلة.

عمليًا، يتم عرض كيفية فتح LSTM بمرور الوقتفي الشكل أدناه:



في البداية، يتم إعطاء قيمة 4.2 للشبكة كمدخلات؛ تم ضبط بوابة الإدخال على 1، لذلك يتم حفظ القيمة الكاملة. ثم بالنسبة للخطوتين التاليتين، يتم ضبط بوابة النسيان على 1. لذلك يتم تخزين جميع المعلومات أثناء هذه الخطوات ولا تتم إضافة أي معلومات جديدة، لأن البوابات مضبوطة على 0. أخيرًا، يتم ضبط بوابة الإخراج على 1 ويتم إنشاء 4.2 وتبقى دون تغيير.

توليد نص باستخدام LSTM

في مولد النص المستند إلى LSTM، قمنا بزيادة طول التسلسل إلى 160 حرفًا مقارنة بالمثال السابق. ومن ثم قمنا بإعادة بناء مجموعات التدريب X وY بالقيمة الجديدة SEQ_LENGTH := 160

```
seq_length = 160
n_seq = int(n_chars / seq_length)

X = np.zeros((n_seq, seq_length, n_vocab))
Y = np.zeros((n_seq, seq_length, n_vocab))

for i in range(n_seq):
        x_sequence = raw_text[i * seq_length : (i + 1) * seq_length]
        x_sequence_ohe = np.zeros((seq_length, n_vocab))
        for j in range(seq_length):
                char = x_sequence[j]
                index = char_to_index[char]
                x_sequence_ohe[j][index] = 1.
        X[i] = x_sequence_ohe
        y_sequence = raw_text[i * seq_length + 1 : (i + 1) * seq_length + 1]
        y_sequence_ohe = np.zeros((seq_length, n_vocab))
        for j in range(seq_length):
                char = y_sequence[j]
                index = char_to_index[char]
                y_sequence_ohe[j][index] = 1.
```

```
        Y[i] = y_sequence_ohe
```

مقارنةً بنموذج RNN السابق، نستخدم نموذجًا بطبقتين متكررتين تحتويان على 800 خلية عصبية وحذف عشوائي باحتمال 0.4:

```
from keras.layers.recurrent import LSTM
batch_size = 100
n_layer = 2
hidden_units = 800
n_epoch= 300
dropout = 0.4
```

نقوم الآن بإنشاء الشبكة وتجميعها:

```
model = Sequential()
model.add(LSTM(hidden_units, input_shape=(None, n_vocab),
return_sequences=True))
model.add(Dropout(dropout))
for i in range(n_layer - 1):
    model.add(LSTM(hidden_units, return_sequences=True))
    model.add(Dropout(dropout))
model.add(TimeDistributed(Dense(n_vocab)))
model.add(Activation('softmax'))
```

بالنسبة للمحسن، نستخدم RMSprop، بمعدل تعلم 0.001:

```
optimizer = keras.optimizers.RMSprop(learning_rate=0.001, rho=0.9,
epsilon=1e-08, decay=0.0)

model.compile(loss= "categorical_crossentropy", optimizer=optimizer)
```

دعونا نلخص نموذج LSTM الذي قمنا ببنائه للتو:

```
model.summary()
```

```
________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
lstm_1 (LSTM) (None, None, 800) 2745600
_________________________________________________________________
dropout_1 (Dropout) (None, None, 800) 0
_________________________________________________________________
lstm_2 (LSTM) (None, None, 800) 5123200
_________________________________________________________________
dropout_2 (Dropout) (None, None, 800) 0
_________________________________________________________________
time_distributed_1 (TimeDist (None, None, 57) 45657
_________________________________________________________________
activation_1 (Activation) (None, None, 57) 0
=================================================================
Total params: 7,914,457
Trainable params: 7,914,457
Non-trainable params: 0
```

يوجد حوالي 8 ملايين معامل للتدريب، وهو ما يقرب من أربعة أضعاف نموذج RNN السابق.

لنبدأ التدريب:

```
model.fit(X, Y, batch_size=batch_size, verbose=1, epochs=n_epoch,
                 callbacks=[ResultChecker(model, 10, 200), checkpoint, early_stop])
```

يكتب المولد نصًا مكونًا من 500 حرف لكل 10 دورات. النتائج التالية خاصة بالدورات 151 و201 و251:

Epoch 151:

```
Epoch                                                                  151/300
19976/19976 [==============================] - 250s 12ms/step - loss:
0.7300
My War and Peace:
ing to the countess. "i have nothing to do with him and i have nothing to
do with the general," said prince andrew.
"i am so sorry for the princess, i am so since he will not be able to say
anything. i saw him long ago. i am so sincerely that i am not to blame for
it. i am sure that something is so much talk about the emperor alexander's
personal attention."
"why do you say that?" and she recognized in his son's presence.
"well, and how is she?" asked pierre.
"the prince is very good to make

Epoch 00151: loss improved from 0.73175 to 0.73003, saving model to
weights/weights_epoch_151_loss_0.7300.hdf5
```

Epoch 201:

```
Epoch 201/300
19976/19976 [==============================] - 248s 12ms/step - loss:
0.6794
My War and Peace:
was all the same to him. he received a story proved that the count had not
yet seen the countess and the other and asked to be able to start a tender
man than the world. she was not a family affair and was at the same time as
in the same way. a few minutes later the count had been at home with his
smile and said:
"i am so glad! well, what does that mean? you will see that you are always
the same."
"you know i have not come to the conclusion that i should like to
send my private result. the prin

Epoch 00201: loss improved from 0.68000 to 0.67937, saving model to
weights/weights_epoch_151_loss_0.6793.hdf5
```

Epoch 251:

```
Epoch 251/300
19976/19976 [==============================] - 249s 12ms/step - loss:
0.6369
My War and Peace:
nd the countess was sitting in a single look on
her face.
"why should you be ashamed?"
"why do you say that?" said princess mary. "why didn't you say a word of
this?" said prince andrew with a smile.
"you would not like that for my sake, prince vasili's son, have you seen
the rest of the two?"
"well, i am suffering," replied the princess with a sigh. "well, what a
delightful norse?" he shouted.
the convoy and driving away the flames of the battalions of the first
day of the orthodox russian
```

```
Epoch 00251: loss improved from 0.63715 to 0.63689, saving model to
weights/weights_epoch_251_loss_0.6368.hdf5
```

أخيرًا ، في الدورة 300 ، توقف التدريب عند الخطأ 0.6001.

بفضل بُنية LSTM ، يستطيع مولد النص كتابة قصة أكثر واقعية وإثارة للاهتمام عن "الحرب والسلام". بالإضافة إلى ذلك ، لا تقتصر شبكات RNN LSTM على توليد الأحرف للنص. يمكنهم التعلم من أي بيانات نصية ، مثل HTML و LaTex وما إلى ذلك.

تصنيف النص متعدد العلامات باستخدام LSTM

في هذا القسم، سنشرح كيفية إنشاء نموذج تصنيف نص بمخرجات متعددة. سنقوم بتطوير نموذج تصنيف النص الذي يحلل تعليقًا نصيًا ويتنبأ بالعديد من العلامات المتعلقة بالتعليق.

تحتوي مجموعة البيانات المستخدمةفي التدريب على ست علامات إخراج لكل تعليق: toxic، severe_toxic، obscene، threat، insult وidentity_hate. يمكن أن ينتمي التعليق إلى كل هذه الفئات أو الفئات الفرعية لهذه الفئات، مما يجعله مسئلة تصنيف متعددة العلامات. يمكنك تنزيل مجموعة البيانات هذه من موقع ويب Kaggle[1] هذا.في هذا المثال، سنستخدم ملف "train.csv" فقط.

أولاً ندخل المكتبات المطلوبة ونحمّل مجموعة البيانات. الكود التالي يدخل المكتبات المطلوبة:

```
from numpy import array
from keras.preprocessing.text import one_hot
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers.core import Activation, Dropout, Dense
from keras.layers import Flatten, LSTM
from keras.layers import GlobalMaxPooling1D
from keras.models import Model
from keras.layers.embeddings import Embedding
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from keras.layers import Input
from keras.layers.merge import Concatenate
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
```

الآن قم بتحميل مجموعة البيانات:

```
toxic_comments = pd.read_csv("train.csv")
```

يعرض الكود التالي ابعاد مجموعة البيانات:

```
print(toxic_comments.shape)
```

```
(159571, 8)
```

[1] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview

كما يتضح، تحتوي مجموعة البيانات على 159،571 سجلًا و8 أعمدة.

باستخدام الأمر التالي، يمكنك رؤية بعض الأمثلة على بيانات الاخراج:

```
toxic_comments.head()
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

في الخطوة التالية، نحذف جميع السجلات التي يحتوي كل صف فيها على قيمة فارغة أو سلسلة فارغة.

```
filter = toxic_comments["comment_text"] != ""
toxic_comments = toxic_comments[filter]
toxic_comments = toxic_comments.dropna()
```

يحتوي العمود comment_text على تعليقات نصية. دعنا نطبع تعليقًا ثم نرى علامات التعليق:

```
print(toxic_comments["comment_text"][168])
```

```
You should be fired, you're a moronic wimp who is too lazy to do research.
It makes me sick that people like you exist in this world.
```

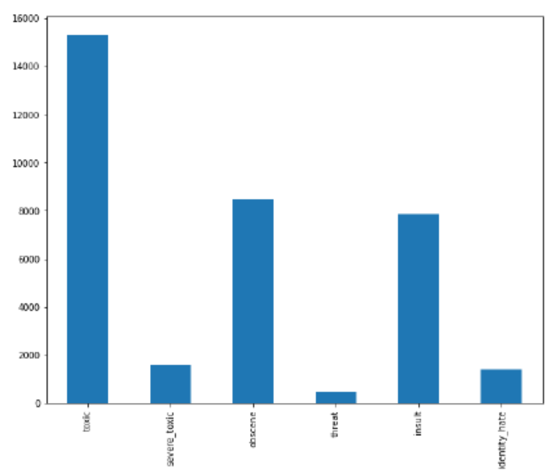
الآن مع الأمر التالي، دعنا نلقي نظرة على العلامات المرتبطة بهذا التعليق:

```
print("Toxic:" + str(toxic_comments["toxic"][168]))
print("Severe_toxic:" + str(toxic_comments["severe_toxic"][168]))
print("Obscene:" + str(toxic_comments["obscene"][168]))
print("Threat:" + str(toxic_comments["threat"][168]))
print("Insult:" + str(toxic_comments["insult"][168]))
print("Identity_hate:" + str(toxic_comments["identity_hate"][168]))
```

```
Toxic:1
Severe_toxic:0
Obscene:0
Threat:0
Insult:1
Identity_hate:0
```

الآن دعنا نوضح عدد التعليقات لكل علامة:

```
toxic_comments_labels = toxic_comments[["toxic", "severe_toxic", "obscene",
"threat", "insult", "identity_hate"]]
fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 10
fig_size[1] = 8
plt.rcParams["figure.figsize"] = fig_size
toxic_comments_labels.sum(axis=0).plot.bar()
```



يمكنك أن ترى أن الفئة "toxic" هي الأكثر شيوعًا. لقد نجحنافي تحليل مجموعة البيانات الخاصة بنا. فيما يلي، سننشئ نموذج مصنف متعدد العلامات لمجموعة البيانات هذه.

بشكل عام، هناك طريقتان لإنشاء نموذج مصنف متعدد العلامات: استخدام طبقة إخراج متصلة بالكامل واستخدام طبقات إخراج متعددة متصلة بالكامل.في الطريقة الأولى، يمكننا استخدام طبقة متصلة بالكامل مع ست مخرجات مع دالة التفعيل sigmoid ووظيفة خطأ الانتروبيا المتقاطع الثنائي. تمثل كل خلية عصبيةفي الطبقة المتصلة بالكامل ناتج إحدى العلامات الست. كما نعلم، ترجع دالة التفعيل sigmoid قيمة بين 0 و1 لكل خلية عصبية. إذا كانت قيمة خرج كل خلية عصبية أكبر من 0.5، فمن المفترض أن التعليق ينتمي إلى الفئة التي يتم تمثيل الخلايا العصبية الخاصة بها.

في الطريقة الثانية، يمكن إنشاء طبقة إخراج متصلة بالكامل لكل علامة.في هذا المثال، تحتاج إلى إنشاء 6 طبقات متصلة بالكامل عند الإخراج، كل طبقة لها دالة sigmoid الخاصة بها.

سنستخدم الطريقة الأولى فقط لمجموعة البيانات هذه وننشئ نموذج مصنف نص متعدد العلامات بطبقة إخراج واحدة. أولاً، نقوم بإنشاء دالة لتنظيف النص:

```
def preprocess_text(sen):
    # Remove punctuations and numbers
    sentence = re.sub('[^a-zA-Z]', ' ', sen)
    # Single character removal
    sentence = re.sub(r"\s+[a-zA-Z]\s+", ' ', sentence)
    # Removing multiple spaces
    sentence = re.sub(r'\s+', ' ', sentence)
    return sentence
```

الخطوة التالية نقوم بإنشاء مجموعة المدخلات والمخرجات الخاصة بنا. إدخال التعليق هو العمود comment_text. نقوم بتخزين جميع التعليقاتفي المتغير X. العلامات أو المخرجات مخزنة بالفعلفي toxic_comments_labels. نستخدم هذه القيم لتخزين المخرجاتفي المتغير y. الكود التالي يوضح هذا:

```
X = []
```

```
sentences = list(toxic_comments["comment_text"])
for sen in sentences:
    X.append(preprocess_text(sen))
y = toxic_comments_labels.values
```

في مجموعة البيانات هذه، لا نحتاج إلى إجراء ترميز one-hot، لأن علامات الإخراج الخاصة بنا قد تم تحويلها بالفعل إلى متجهات ترميز one-hot.في الخطوة التالية، نقسم بياناتنا إلى مجموعات بيانات تدريب واختبار:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=42)
```

بعد ذلك، نحتاج إلى تحويل مدخلاتنا إلى متجهات عددية. لذا قبل أن ننتقل إلى هذا المثال، دعنا نتعلم المزيد عن **تضمين الكلمات (word embedding)**. لا تستطيع خوارزميات التعلم العميق فهم البيانات النصية الأولية، لذلك يجب تحويل النص حتى تتمكن الشبكة من فهمه ومعالجته. إن تضمين الكلمات هو طريقة لتمثيل الكلمات التي تهدف إلى التمثيل المعنوي للكلماتفي شكل متجهات حقيقية، حيث يتم تمثيل الكلمات ذات المعاني والسياقات المتشابهة بواسطة متجهات مماثلة. هذه المتجهات العددية أصغر من الأساليب الإحصائيةفي معالجة اللغة الطبيعية لتحويل الكلمات إلى أرقام. أيضًا، هذه المتجهات العددية، إذا كانت مدربة جيدًا، لديها القدرة على إظهار الروابط الدلالية والنحوية بين الكلمات. يعد تضمين الكلمات الحجر الاساس للعديد من وظائف معالجة اللغة الطبيعية التي تستخدم التعلم العميق. يمكن تحقيق تضمين الكلمات للنصوص من طريقتين مختلفتين.في الطريقة الأولى، يتم تعليمهمفي وقت واحد مع المهمة الرئيسية أثناء تدريب الشبكة.في هذه الطريقة، يتم أولاً إنشاء القيم الرقمية للمتجهات بشكل عشوائي ثم أثناء التدريب يتم تحديث هذه القيم من خلال التحسين مثل طبقات الشبكة الأخرى. الطريقة الثانية هي من خلال التدريب باستخدام خوارزميات خاصة مثل fasttext وGloVe على مجموعة كبيرة من البيانات النصية واستخدام الأوزان التي تم الحصول عليها من هذه الخوارزميات.

الآن دعنا نحول مدخلات النص إلى متجهات مضمنة:

```
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(X_train)
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)
vocab_size = len(tokenizer.word_index) + 1
maxlen = 200
X_train = pad_sequences(X_train, padding='post', maxlen=maxlen)
X_test = pad_sequences(X_test, padding='post', maxlen=maxlen)
```

سنستخدم مضمن الكلمات GloVe لتحويل مدخلات النص إلى نظيراتها الرقمية. أدخل الكود أدناه لتنزيله:

```
!wget http://nlp.stanford.edu/data/glove.6B.zip
!unzip glove*.zip
```

لاستخدام GloVe، نقوم بما يلي:

```
from numpy import array
from numpy import asarray
from numpy import zeros

embeddings_dictionary = dict()

glove_file = open('glove.6B.100d.txt', encoding="utf8")

for line in glove_file:
    records = line.split()
    word = records[0]
    vector_dimensions = asarray(records[1:], dtype='float32')
    embeddings_dictionary[word] = vector_dimensions
glove_file.close()

embedding_matrix = zeros((vocab_size, 100))
for word, index in tokenizer.word_index.items():
    embedding_vector = embeddings_dictionary.get(word)
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector
```

ثم نقوم بإنشاء نموذجنا بالكود التالي. سيحتوي نموذجنا على طبقة إدخال ، وطبقة تضمين ، وطبقة LSTM بها 128 خلية عصبية وطبقة إخراج بها 6 خلايا عصبية ، لأن لدينا 6 علامات في الإخراج.

```
deep_inputs = Input(shape=(maxlen,))
embedding_layer = Embedding(vocab_size, 100, weights=[embedding_matrix],
trainable=False)(deep_inputs)
LSTM_Layer_1 = LSTM(128)(embedding_layer)
dense_layer_1 = Dense(6, activation='sigmoid')(LSTM_Layer_1)
model = Model(inputs=deep_inputs, outputs=dense_layer_1)

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
```

دعونا نطبع ملخص النموذج:

```
print(model.summary())
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 200)]             0

 embedding (Embedding)       (None, 200, 100)          14824300

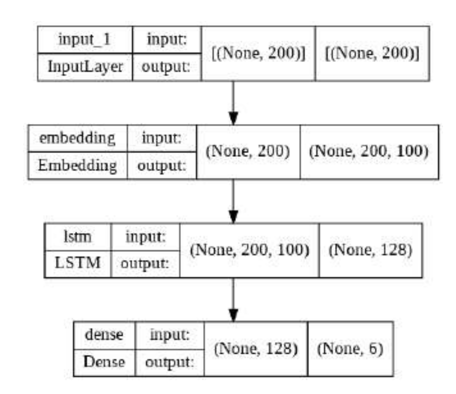
 lstm (LSTM)                 (None, 128)               117248

 dense (Dense)               (None, 6)                 774

=================================================================
Total params: 14,942,322
Trainable params: 118,022
Non-trainable params: 14,824,300
```

يمكنك تصوير بُنية الشبكة العصبية الخاصة بك باستخدام الكود التالي:

```
from keras.utils.vis_utils import plot_model
plot_model(model, to_file='model_plot4a.png', show_shapes=True,
show_layer_names=True)
```



يمكنك أن ترى من الشكل أعلاه أن طبقة الإخراج تتكون من طبقة واحدة متصلة بالكامل بها 6 خلايا عصبية. الآن دعنا ندرب النموذج:

```
history = model.fit(X_train, y_train, batch_size=128, epochs=5, verbose=1,
validation_split=0.2)
```

```
Epoch 1/5
798/798 [==============================] - 20s 17ms/step - loss: 0.1193 - acc: 0.9684 - val_loss:
0.0739 - val_acc: 0.9941
Epoch 2/5
798/798 [==============================] - 13s 17ms/step - loss: 0.0643 - acc: 0.9927 - val_loss:
0.0599 - val_acc: 0.9943
Epoch 3/5
798/798 [==============================] - 13s 17ms/step - loss: 0.0572 - acc: 0.9938 - val_loss:
0.0573 - val_acc: 0.9935
Epoch 4/5
798/798 [==============================] - 13s 17ms/step - loss: 0.0548 - acc: 0.9939 - val_loss:
0.0566 - val_acc: 0.9943
Epoch 5/5
798/798 [==============================] - 14s 17ms/step - loss: 0.0523 - acc: 0.9940 - val_loss:
0.0542 - val_acc: 0.9942
```

لنقم الآن بتقييم النموذج في مجموعة الاختبار:

```
score = model.evaluate(X_test, y_test, verbose=1)
print("Test Score:", score[0])
print("Test Accuracy:", score[1])
```

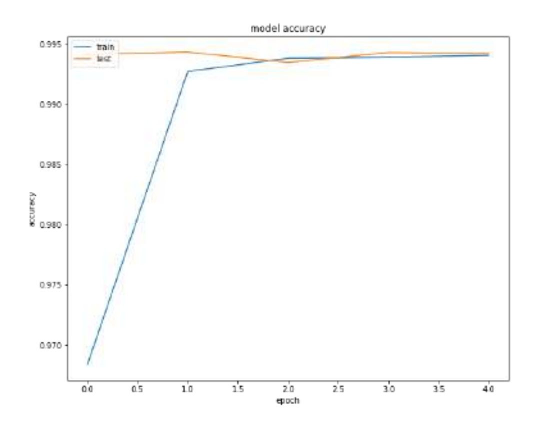
```
998/998 [==============================] - 6s 6ms/step - loss: 0.0529 - acc: 0.9938
Test Score: 0.05285229906439781
Test Accuracy: 0.9937959909439087
```

لقد حقق النموذج دقة تصل إلى 99٪ في مجموعة الاختبار وهو أمر ممتاز. أخيرًا ، قمنا برسم قيم الخطأ والدقة لمجموعات التدريب والاختبار لمعرفة ما إذا كان النموذج قد أدى إلى overfitting.

```
import matplotlib.pyplot as plt

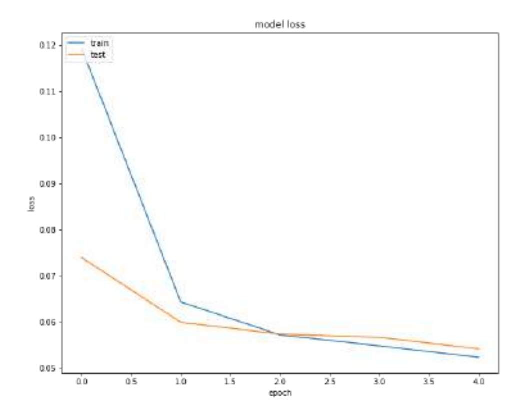
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])

plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
```




```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
```



كما ترون في الصور أعلاه ، لم يؤدي النموذج إلى overfitting.

تحليل العاطفة مع LSTM

مع ظهور أدوات الويب 2 وظهور وسائل التواصل الاجتماعي، أصبحت حياة المجتمعات البشرية اليوم متشابكة بشدة. وقد أدى ذلك إلى إنتاج كميات هائلة من البيانات من قبل مستخدمي هذه الوسائط. تعد البيانات النصية من أكثر البيانات استخدامًا والتي يمكن استخدامها للحصول على معلومات مهمة حول مواضيع مختلفة. تؤدي وسائل التواصل الاجتماعي بأشكالها المختلفة مثل المنتديات والمدونات ومواقع التعليقات وما إلى ذلك، إلى إنتاج كميات كبيرة من البيانات بشكل يومي. هذه البياناتفي شكل تعليقات وانتقادات ووجهات نظر حول الخدمات والشركات والمنظمات والأحداث والأشخاص والقضايا والموضوعات. التعليقات التي يقدمها المستخدمون على الشبكات الاجتماعية مهمة جداً وعملية.في متجر على الإنترنت، يمكن أن

تعكس الآراء ووجهات النظر المختلفة حول منتج ما مستوى رضا العملاء وجودته، والتي يمكن أن تكون دليلًا رائعًا للمشترين الآخرين. ليس من السهل تصنيف وتنظيم هذا الكم الهائل من الآراء يدويًا حول موضوع معين. لذلك، أدت الحاجة إلى نظام آلي لجمع الآراء إلى ظهور مجال بحث جديد يسمى **تحليل المشاعر (Sentiment analysis)**. تحليل المشاعر هو مجال دراسي هدفه الرئيسي تحديد واستخراج وتصنيف المشاعر والآراء والمواقف والأفكار والأحكام والنقد ووجهات النظر تجاه الكائنات والمنظمات والأحداث وما إلى ذلك دون تفاعل بشريفي شكل إيجابي أو سلبي أو فئات محايدة.

النهجان المختلفان اللذان يستخدمهما الباحثون لتصنيف المشاعر في النص هما نهجان يعتمدان على المفردات و تعلم الآلة. يمكن اعتبار نهج آخر مزيجًا من الاثنين. يركز النهج القائم على المفردات على استخراج الكلمات أو العبارات التي يمكن أن توجه عملية التصنيف في اتجاه دلالي محدد. كل كلمة لها عبء دلالي محدد يتم استخراجه من خلال مسرد الكلمات ذات الشحنات العاطفية الإيجابية والسلبية التي تم تسجيلها بالفعل. عن طريق إضافة نقاط الحمل العاطفي للكلمات أو حساب عدد الكلمات ذات الشحنات الموجبة والسالبة ، يتم الحصول على القطبية العامة للجملة. يمكن تدريس نهج التعلم الآلي واستخدامه بعدة طرق لتحليل المشاعر. في وضع التعلم الخاضع للإشراف ،يتم الإشراف على نموذج التدريب بمجموعة بيانات تدريب مصنفة مسبقًا ليكون قادرًا على التعلم والتصرف بشكل مشابه للبيانات المدربة في مواجهة البيانات غير المرئية.

في السنوات الأخيرة ، أثبت الباحثون على نطاق واسع أن نماذج التمثيل القائم على التعلم العميق أكثر فاعلية في القضايا المتعلقة بتصنيف المشاعر. يرجع اعتماد مناهج التعلم العميق في تحليل المشاعر إلى القدرة العالية جدًا لنماذج التعلم العميق على تعلم الميزات تلقائيًا ، والتي يمكن أن تحقق دقة وأداء أفضل. في العديد من مجالات معالجة اللغة الطبيعية ، أدى استخدام التعلم العميق إلى نتائج تتجاوز تلك المستخدمة سابقًا بواسطة التعلم الآلي والأساليب الإحصائية.

الآن بعد أن أصبحت على دراية بتحليل المشاعر ، فلنقم ببناء نموذج لتحليل المشاعر في مجال مراجعة الأفلام بمساعدة شبكة LSTM. لهذا التنفيذ ، نستخدم مجموعة بيانات IMDB. تتمثل ميزة مجموعة البيانات هذه في أنها مضمنة بالفعل في مكتبة مجموعة بيانات Keras.

أولاً ، نقوم بتحميل مجموعة البيانات عبر الكود التالي:

```
from keras.datasets import imdb
top_words = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=top_words)
```

يقسم الكود أعلاه مجموعة البيانات إلى مجموعتين تدريب واختبار في نفس الوقت الذي يتم فيه تحميل أفضل 5000 كلمة لكل مراجعة. دعنا نلقي نظرة على مجموعة البيانات:

```
X_train
```

```
array([list([1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5,
25, 100, 43, 838, 112, 50, 670, 22665, 9, 35, 480, 284, 5, 150, 4, 172, 112, 167, 21631, 336, 385,
39, 4, 172, 4536, 1111, 17, 546, 38, 13, 447, 4, 192, 50, 16, 6, 147, 2025, 19, 14, 22, 4, 1920,
4613, 469, 4, 22, 71, 87, 12, 16, 43, 530, 38, 76, 15, 13, 1247, 4, 22, 17, 515, 17, 12, 16, 626,
18, 19193, 5, 62, 386, 12, 8, 316, 8, 106, 5, 4, 2223, 5244, 16, 480, 66, 3785, 33, 4, 130, 12,
16, 38, 619, 5, 25, 124, 51, 36, 135, 48, 25, 1415, 33, 6, 22, 12, 215, 28, 77, 52, 5, 14, 407,
16, 82, 10311, 8, 4, 107, 117, 5952, 15, 256, 4, 31050, 7, 3766, 5, 723, 36, 71, 43, 530, 476, 26,
400, 317, 46, 7, 4, 12118, 1029, 13, 104, 88, 4, 381, 15, 297, 98, 32, 2071, 56, 26, 141, 6, 194,
7486, 18, 4, 226, 22, 21, 134, 476, 26, 480, 5, 144, 30, 5535, 18, 51, 36, 28, 224, 92, 25, 104,
4, 226, 65, 16, 38, 1334, 88, 12, 16, 283, 5, 16, 4472, 113, 103, 32, 15, 16, 5345, 19, 178, 32]),
       list([1, 194, 1153, 194, 8255, 78, 228, 5, 6, 1463, 4369, 5012, 134, 26, 4, 715, 8, 118,
       ...,
       list([1, 1446, 7079, 69, 72, 3305, 13, 610, 930, 8, 12, 582, 23, 5, 16, 484, 685, 54, 349,
11, 4120, 2959, 45, 58, 1466, 13, 197, 12, 16, 43, 23, 21469, 5, 62, 30, 145, 402, 11, 4131, 51,
575, 32, 61, 369, 71, 66, 770, 12, 1054, 75, 100, 2198, 8, 4, 105, 37, 69, 147, 712, 75, 3543, 44,
257, 390, 5, 69, 263, 514, 105, 50, 286, 1814, 23, 4, 123, 13, 161, 40, 5, 421, 4, 116, 16, 897,
13, 40691, 40, 319, 5872, 112, 6700, 11, 4803, 121, 25, 70, 3468, 4, 719, 3798, 13, 18, 31, 62,
40, 8, 7200, 4, 29455, 7, 14, 123, 5, 942, 25, 8, 721, 12, 145, 5, 202, 12, 160, 580, 202, 12, 6,
52, 58, 11418, 92, 401, 728, 12, 39, 14, 251, 8, 15, 251, 5, 21213, 12, 38, 84, 80, 124, 12, 9,
23]),
       list([1, 17, 6, 194, 337, 7, 4, 204, 22, 45, 254, 8, 106, 14, 123, 4, 12815, 270, 14437, 5,
16923, 12255, 732, 2098, 101, 405, 39, 14, 1034, 4, 1310, 9, 115, 50, 305, 12, 47, 4, 168, 5, 235,
7, 38, 111, 699, 102, 7, 4, 4039, 9245, 9, 24, 6, 78, 1099, 17, 2345, 16553, 21, 27, 9685, 6139,
5, 29043, 1603, 92, 1183, 4, 1310, 7, 4, 204, 42, 97, 90, 35, 221, 109, 29, 127, 27, 118, 8, 97,
12, 157, 21, 6789, 85010, 9, 6, 66, 78, 1099, 4, 631, 1191, 5, 2642, 272, 191, 1070, 6, 7585, 8,
2197, 70907, 10755, 544, 5, 383, 1271, 848, 1468, 12183, 497, 16876, 8, 1597, 8778, 19280, 21, 60,
27, 239, 9, 43, 8368, 209, 405, 10, 10, 12, 764, 40, 4, 248, 20, 12, 16, 5, 174, 1791, 72, 7, 51,
6, 1739, 22, 4, 204, 131, 9])],
      dtype=object)
```

إذا نظرت إلى البيانات أعلاه ، ستلاحظ أن البيانات قد تمت معالجتها بالفعل. جميع الكلمات مكتوبة بأعداد صحيحة ، وتمثل الأعداد الصحيحة الكلمات مرتبة حسب عدد تكرارها. على سبيل المثال ، تمثل 4 الكلمة الرابعة الأكثر استخدامًا ، و 5 تمثل الكلمة الخامسة الأكثر استخدامًا ، وهكذا. العدد الصحيح 1 محجوز لعلامة البداية ، والعدد الصحيح 2 محجوز لكلمة غير معروفة ، و 0 للحشو. إذا كنت تريد إلقاء نظرة على المراجعات بنفسك ومعرفة ما كتبه الأشخاص ، يمكنك عكس هذا الاتجاه أيضًا:

```python
word_index = imdb.get_word_index() # get {word : index}
index_word = {v : k for k,v in word_index.items()} # get {index : word}
index = 1
print(" ".join([index_word[idx] for idx in x_train[index]]))
print("positve" if y_train[index]==1 else "negetive")
```

```
the thought solid thought senator do making to is spot nomination assumed while he of
jack in where picked as getting on was did hands fact characters to always life
thrillers not as me can't in at are br of sure your way of little it strongly random
to view of love it so principles of guy it used producer of where it of here icon film
of outside to don't all unique some like of direction it if out her imagination below
keep of queen he diverse to makes this stretch stefan of solid it thought begins br
senator machinations budget worthwhile though ok brokedown awaiting for ever better
were lugia diverse for budget look kicked any to of making it out bosworth's follows
for effects show to show cast this family us scenes more it severe making senator to
levant's finds tv tend to of emerged these thing wants but fuher an beckinsale cult as
it is video do you david see scenery it in few those are of ship for with of wild to
one is very work dark they don't do dvd with those them
negetive
```

نظرًا لاختلاف المراجعات من حيث الطول ، فإننا نريد تقسيم كل مراجعة إلى أول 500 كلمة. نحتاج إلى عينات نصية بنفس الطول حتى نتمكن من إدخالها في شبكتنا العصبية. إذا كانت المراجعات أقصر من 500 كلمة ، نضيفها بصفر.

يعد Keras رائعًا لهذا ، لأنه يوفر مجموعة من إجراءات المعالجة المسبقة التي يمكن إجراؤها بسهولة بالنسبة لنا:

```
word_index = imdb.get_word_index() # get {word : index}
index_word = {v : k for k,v in word_index.items()} # get {index : word}
index = 1
print(" ".join([index_word[idx] for idx in x_train[index]]))
print("positve" if y_train[index]==1 else "negetive")
```

لفهم أفضل ، دعنا نختار عينة من البيانات بشكل عشوائي ونرى ما فعله الكود أعلاه:

```
X_train[125]
```

```
array([    0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     1,    11,
           6,    58,    54,     4, 14537,     5,  6495,     4,  2351,
        1630,    71, 13202,    23,    26, 20094, 40865,    34,    35,
        9454,  1680,     8,  6681,   692,    39,    94,   205,  6177,
         712,   121,     4, 18147,  7037,   406,  2657,     5,  2189,
       61778,    26, 23906, 11420,     6,   708,    44,     4,  1110,
         656,  4667,   206,    15,   230, 13781,    15,     7,     4,
        4847,    36,    26, 54759,   238,   306,  2316,   190,    48,
          25,   181,     8,    67,     6,    22,    44,    15,   353,
        1659, 84675,  3048,     4,  9818,   305,    88, 11493,     9,
          31,     7,     4,    91, 12789, 53410,  3106,   126,    93,
          40,   670,  8759, 41931,     6,  6951,     4,   167,    47,
         623,    23,     6,   875,    29,   186,   340,  4447,     7,
           5, 44141,    27,  5485,    23,   220,   175,  2122,    10,
          10,    27,  4847,    26,     6,  5261,  2631,   604,     7,
        2118, 23310, 36011,  5350,    17,    48,    29,    71, 12129,
          18,  1168, 38886, 33829,  1918, 31235,  3255,  9977, 31537,
        9248,    40,    35,  1755,   362,    11,     4,  2370,  2222,
          56,     7,     4, 23052,  2489,    39,   609, 82401, 48583,
           6,  3440,   655,   707,  4198,  3801,    37,  4486,    33,
         175,  2631,   114,   471,    17,    48,    36,   181,     8,
          30,  1059,     4,  3408,  5963,  2396,     6,   117,   128,
          21,    26,   131,  4218,    11, 20663,  3826, 14524,    10,
          10,    12,     9,   614,     8,    97,     6,  1393,    22,
          44,   995,    84, 21800,  5801,    21,    14,     9,     6,
        4953,    22,    44,   995,    84,    93,    34,    84,    37,
         104,   507, 11076,    37,    26,   662,   180,     8,     4,
        5075,    11,   882,    71,    31,     8, 39022, 36011, 31537,
           5, 48583,    19,  1240, 31800,  1806, 11521,     5,  7863,
       28281,     4,   959,    62,   165,    30,     8,  2988,     4,
        2772,  1500,     7,     4,    22,    24,  2358,    12,    10,
          10, 22993,   238,    43,    28,   188,   245,    19,    27,
         105,     5,   687,    48,    29,   562,    98,   615,    21,
          27,  8500,     9,    38,  2797,     4,   548,   139,    62,
        9343,     6, 14053,   707,   137,     4,  1205,     7,     4,
        6556,     9,    53,  2797,    74,     4,  6556,   410,     5,
          27,  5150,     8,    79,    27,   177,     8,  3126,    19,
          33,   222,    49, 22895,     7, 14090,   406,  5424,    38,
        4144,    15,    12,     9,   165,  2268,     8,   106,   318,
         760,   215,    30,    93,   133,     7, 31537,    38,    55,
          52,    11, 26149, 17310,  1080, 24192, 68007,    29,     9,
        6248,    78,   133,    11,     6,   239,    15,     9,    38,
         230,   120,     4,   350,    45,   145,   174,    10,    10,
       22993,    47,    93,    49,   478,   108,    21,    25,    62,
         115,   482,    12,    39,    14,  2342,   947,     6,  6950,
           8,    27,   157,    62,   115,   181,     8,    67,   160,
           7,    27,   108,   103,    14,    63,    62,    30,     6,
```

```
         87,    902,   2152,   3572,      5,      6,    619,    437,      7,
          6,   4616,    221,    819,     31,    323,     46,      7,    747,
          5,    198,    112,     55,   3591], dtype=int32)
```

كما ترى أعلاه ، نظرًا لأن سجل الطول هذا كان أقل من 500 كلمة ، يتم وضع رقم 0 أمامه ليكون سجل الطول 500.

من المثير للدهشة أن عمل معالجة البيانات لدينا قد اكتمل ويمكننا الآن البدء في بناء النموذج:

```
from keras.models import Sequential
from keras.layers import Embedding
from keras.layers import LSTM, Dense
embedding_vector_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',
metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential_2"

_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding (Embedding)       (None, 500, 32)           160000    
                                                                 
 lstm (LSTM)                 (None, 100)               53200     
                                                                 
 dense (Dense)               (None, 1)                 101       
                                                                 
=================================================================
Total params: 213,301
Trainable params: 213,301
Non-trainable params: 0
_________________________________________________________________
None
```

كما ذكرنا سابقًا ، هناك طريقتان لتضمين الكلمات. في المثال السابق استخدمنا مضمن الكلمات المدرب مسبقا. في هذا المثال نستخدم طبقة embedding. تتعلم طبقة Embedding تضمين كلمة من مجموعة البيانات.

حان الوقت الآن لتدريب النموذج:

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=5,
batch_size=32)
```

```
Epoch 1/5
782/782 [==============================] - 82s 103ms/step - loss: 0.4681 - accuracy:
0.7815 - val_loss: 0.3702 - val_accuracy: 0.8430
Epoch 2/5
782/782 [==============================] - 79s 101ms/step - loss: 0.3340 - accuracy:
0.8618 - val_loss: 0.3984 - val_accuracy: 0.8299
Epoch 3/5
782/782 [==============================] - 80s 102ms/step - loss: 0.2669 - accuracy:
0.8954 - val_loss: 0.3272 - val_accuracy: 0.8657
Epoch 4/5
782/782 [==============================] - 80s 102ms/step - loss: 0.2259 - accuracy:
0.9117 - val_loss: 0.3122 - val_accuracy: 0.8713
Epoch 5/5
782/782 [==============================] - 79s 101ms/step - loss: 0.1912 - accuracy:
0.9270 - val_loss: 0.3399 - val_accuracy: 0.8604
```

بعد الانتهاء من تدريب النموذج ، حان الوقت لتقييم أداء النموذج:

```
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Accuracy: 86.04%
```

كما يتضح، فقد تمكن النموذج من تحقيق دقة تبلغ حوالي 86٪ ، وهو أمر ممتاز بالنظر إلى المشكلة الصعبة. ومع ذلك ، فإن هذا النموذج ليس أفضل نموذج ممكن كتابته. كتمرين، يمكنك رؤية النتائج من خلال تجربة معاملات فائقة مختلفة وبناء نموذج عالي الأداء. أيضًا ، يمكنك رسم مخطط الخطأ والدقة للنماذج ومعرفة ما إذا كانت النماذج تؤدي إلى overfitting. فيما يتعلق بالنموذج أعلاه ، ما رأيك في بالضبط الزائد overfitting ؟

خلاصة الفصل

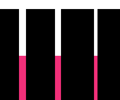
- **تقضي الشبكات العصبية المتكررة على أوجه القصور في الشبكات العصبية أمامية التغذية.**
- **شبكات RNN البسيطة ليست قادرة على تعلم التبعيات طويلة المدى.**
- **يمكن لـ LSTM تكوين تبعيات طويلة المدى بسبب وجود خلية ذاكرة خاصة في هيكلها.**

اختبار

كم عدد البوابات التي تمتلكها LSTM وما هو دور كل منها؟

6

شبكات الخصومة التوليدية



اهداف التعليم:

- الفرق بين نموذج المولد ونموذج المميز.
- التعرف على شبكات الخصومة التوليدية.
- التدريب على الخصومة.
- توليد أرقام مكتوبة بخط اليد باستخدام شبكة الخصومة التوليدية.

المقدمة

في هذا الفصل، نعتزم تقديم مجموعة من نماذج المولد بناءً على بعض مفاهيم نظرية اللعبة. السمة الرئيسية الخاصة بهم هي طريقة تدريب معادية (خصومة) تهدف إلى معرفة التمييز بين العينات الحقيقية والمزيفة من خلال مكون واحد، وفي الوقت نفسه، ينتج مكون آخر عينات أكثر فأكثر تشابهًا مع عينات التدريب. باختصار،في هذه الشبكات، ينتج أحدهم، والآخر يدون الملاحظات، وبالتعاون التام، يحققون نتائج جيدة جدًا.

ما هو النموذج المولد؟

بشكل عام، هناك نوعان رئيسيان من النماذج في التعلم الآلي: **نموذج المولد (generative model) ونموذج المميز(discriminative model)**. يحاول نموذج المميز ، كما يوحي اسمه ، فصل البيانات بين فئتين أو أكثر. بشكل عام ، تركز نماذج المميز على التنبؤ بفئات البيانات وفقًا لخصائصها. على العكس من ذلك، لا يحاول نموذج المولد تعيين سمات للفئات ، ولكنه يولد سمات موجودة في فئة معينة. هناك طريقة سهلة للتمييز بين نموذج المولد ونموذج المميز:

- يركز نموذج المولد على نمذجة توزيع البيانات.
- يهتم نموذج المميز بإيجاد حدود أو قواعد لفصل البيانات.

التعريف 1.6 نموذج المولد

يصف نموذج المولد كيفية إنشاء مجموعة بيانات بناءً على نموذج احتمالي.

تعد نماذج المولد أداة قوية لفحص توزيع البيانات أو تقدير كثافة مجموعات البيانات. تتبع نماذج المولد التعلم غير الخاضع للإشراف الذي يكتشف تلقائيًا الأنماط أو المخالفاتفي البيانات التي يتم تحليلها. يساعد هذافي إنشاء بيانات جديدة تشبه إلى حد كبير مجموعة البيانات الأصلية. على وجه التحديد، الهدف من نماذج المولد هو معرفة التوزيع الفعلي للبياناتفي مجموعة التدريب، لإنشاء نقاط بيانات جديدة مع بعض التعديلات.

الغرض الرئيسي من جميع أنواع نماذج المولد هو معرفة التوزيع الفعلي لبيانات مجموعة التدريب بحيث يمكن إنشاء نقاط بيانات جديدة مع التعديلات. لكن لا يمكن للنموذج أن يتعلم التوزيع الدقيق لبياناتنا، لذلك قمنا بنمذجة توزيع مشابه لتوزيع البيانات الحقيقية. للقيام بذلك، نستخدم معرفة الشبكة العصبية للتعلم الدالي الذي يمكنه تقريب توزيع النموذج إلى التوزيع الفعلي.

يمكن تعريف نماذج المولد على أنها فئة من النماذج التى تهدف إلى معرفة كيفية إنشاء عينات جديدة تبدو وكأنها من نفس مجموعة بيانات التدريب. أثناء مرحلة التدريب، يحاول النموذج المولد حل مشكلة تقدير الكثافة (Density estimation). فى تقدير

الكثافة، يتعلم النموذج إجراء تقدير أقرب ما يمكن إلى دالة كثافة الاحتمال غير المرئي. الأهم من ذلك، يجب أن يكون نموذج المولد قادرًا على تكوين مثيلات جديدة للتوزيع، وليس مجرد نسخ الموجودة.

كانت نماذج المولد في طليعة التعلم العميق غير الخاضع للإشراف على مدار العقد الماضي. هذا لأنها توفر طريقة فعالة للغاية لتحليل وفهم البيانات غير المصنفة.

نماذج المولد ومستقبل الذكاء الاصطناعي؟

هناك ثلاثة أسباب عامة تجعل من الممكن اعتبار نماذج المولد مفتاحًا لإطلاق شكل أكثر تعقيدًا من الذكاء الاصطناعي يتجاوز ما يمكن أن تحققه نماذج المميز.

- **أولاً** ، من وجهة النظر النظرية وحدها ، لا يجب أن نشعر بالرضا عن القدرة الفائقة على تصنيف البيانات فحسب ، بل يجب أيضًا السعي إلى فهم أكثر اكتمالاً لكيفية إنشاء البيانات في المقام الأول. مما لا شك فيه أن حل هذه المشكلة أصعب بكثير من الطرق المقارنة. ومع ذلك ، كما سنرى ، يمكن أيضًا استخدام العديد من نفس الأساليب التي أدت إلى تطوير النمذجة التمييزية (مثل التعلم العميق) بواسطة نماذج المولد.
- **ثانيًا** ، من المرجح أن تكون النمذجة التوليدية أكثر أهمية وفعالية من أي شيء آخر في توجيه التطورات المستقبلية في مجالات أخرى من التعلم الآلي ، مثل التعلم المعزز (Reinforcement learning). على سبيل المثال ، يمكننا استخدام التعلم المعزز لتدريب الروبوت على المشي في منطقة معينة. النهج العام هو بناء محاكاة حاسوبية من الأرض ثم إجراء العديد من التجارب التي يحاول فيها الوكيل استراتيجيات مختلفة. بمرور الوقت ، يتعرف الوكيل على الاستراتيجيات الأكثر نجاحًا من غيرها وبالتالي يتحسن تدريجيًا. المشكلة الشائعة في هذا النهج هي أن فيزياء البيئة غالبًا ما تكون معقدة للغاية ويجب حسابها في كل خطوة لإعادة المعلومات إلى الوكيل ليقرر خطوته التالية. ومع ذلك ، إذا تمكن الوكيل من محاكاة بيئته من خلال نموذج توليدي ، فلن يحتاج إلى تجربة الإستراتيجية في محاكاة الكمبيوتر أو في العالم الحقيقي ، ولكن يمكنه التعلم في بيئته التخيلية.
- **أخيرًا** ، إذا أردنا حقًا أن نقول إننا بنينا آلة اكتسبت شكلاً من أشكال الذكاء يمكن مقارنته بالذكاء البشري ، فيجب أن تكون النمذجة التوليدية بالتأكيد جزءًا من الحل. أحد أفضل الأمثلة على نموذج المولد في العالم الحقيقي هو الشخص الذي يقرأ هذا الكتاب. توقف لحظة لتفكر في ماهية نموذج المولد المذهل الذي أنت عليه. يمكنك أن تغمض عينيك وتتخيل شكل الفيل من كل زاوية ممكنة. يمكنك تخيل عدد من النهايات المختلفة والمقبولة لبرنامجك التلفزيوني المفضل ، ويمكنك أيضًا التخطيط لأسبوعك والعمل فيه

من خلال العمل في المستقبل في عقلك. تُظهر نظرية علم الأعصاب الحالية أن فهمنا للواقع ليس نموذجًا مميزًا معقدًا للغاية يعمل على مدخلاتنا الحسية لتوليد تنبؤات بما نشهده ، بل إنه نموذج مثمر يتم تدريبه منذ الولادة لتوليد محاكاة لمحيطنا تتناسب تمامًا مع المستقبل.

شبكة الخصومة التوليدية (GAN)

فكرفي كيفية التعلم. أنت تحاول شيئًا ما وتحصل على ردود الفعل. قمت بتعديل استراتيجيتك وحاول مرة أخرى. قد تأخذ التعليقات شكل النقد أو الألم أو الكسب. قد يأتي من حكمك على مدى جودة أدائك. ومع ذلك، غالبًا ما تكون التعليقات الأكثر فائدة هي التعليقات الواردة من شخص آخر، لأنها ليست مجرد رقم أو شعور، ولكنها تقييم ذكي لمدى نجاحكفي ذلك.

عندما يتم تدريب جهاز كمبيوتر على مهمة ما، عادةً ما يقدم البشر ملاحظاتفي شكل معاملات أو خوارزميات. تكون هذه الملاحظات أسهل عندما تكون مهمة بسيطة مثل تعلم ضرب رقمين. يمكنك بسهولة إخبار الكمبيوتر بدقة كيف حدث خطأ. ومع ذلك، مع مهمة أكثر تعقيدًا، مثل إنشاء صورة قطة، يصبح تقديم الملاحظات أكثر صعوبة. هل الصورة ضبابية، هل تبدو أشبه بالكلب أم تبدو كشيء على الإطلاق؟ يمكن تنفيذ الإحصائيات المعقدة، لكن من الصعب تسجيل كل التفاصيل التي تجعل الصورة حقيقية. يمكن للمرء أن يقدر، لأن لدينا الكثير من الخبرةفي تقييم المدخلات المرئية، لكننا بطيئون نسبيًا ويمكن أن تكون تقييماتنا ذاتية للغاية. بدلاً من ذلك، يمكننا تدريب شبكة عصبية لتعلم مهمة التمييز بين الصور الحقيقية والمنتجة. بعد ذلك، يمكن إعطاء مولد الصور (الشبكة العصبية) والمميز الفرصة للتعلم من بعضهما البعض والتحسين بمرور الوقت. الشبكتان اللتان تلعبان هذه اللعبة هما الخصومة التوليدية.

شبكات الخصومة التوليدية(Generative Adversarial Networks) ، أو اختصارًا **GAN**، هي فئة من تقنيات التعلم الآلي التي تتكون من نموذجين مدرَّبينفي وقت واحد: مولد مُدرَّب على إنشاء بيانات مزيفة ومميز مُدرَّب. للتمييز بين البيانات المزيفة والعينات الحقيقية. تشير كلمة "**توليد**" إلى الغرض العام للنموذج: إنشاء بيانات جديدة. تعتمد البيانات التي تتعلم GAN على توليدها على اختيار مجموعة التدريب. على سبيل المثال، إذا أردنا أن تجمع GAN بين صور تشبه دافنشي، فإننا نستخدم قاعدة بيانات Da Vinci Art التدريبية.

يشير مصطلح **الخصومة** (**adversarial**) إلى منافسة ديناميكية ومرحة بين النموذجين اللذين يشكلان إطار عمل GAN: **المولد والمميز**. الهدف من المولد هو إنشاء عينات لا يمكن تمييزها عن البيانات الفعليةفي مجموعة التدريب.في مثالنا، هذا يعني إنتاج لوحات تشبه لوحات دافنشي تمامًا. الغرض من المميز هو التمييز بين العينات المزيفة التي ينتجها المولد من العينات الحقيقية التي تم الحصول عليها من مجموعة بيانات التدريب.في مثالنا، تلعب المميز دور خبير فني يقيم

أصالة اللوحات التي يعتقد أنها تنتمي إلى دافنشي. تحاول الشبكتان باستمرار خداع بعضهما البعض: فكلما كان المولد أفضل في إنشاء بيانات واقعية، يجب أن يكون الفارق أفضل في التمييز بين العينات الحقيقية والعينات المزيفة.

أخيرًا، تشير كلمة الشبكات العصبية إلى فئة نماذج التعلم الآلي المستخدمة بشكل شائع لتمثيل المولد والمميز. اعتمادًا على مدى تعقيد تنفيذ GAN، يمكن أن تتراوح من الشبكات العصبية امامية التغذية البسيطة إلى الشبكات العصبية الالتفافية أو حتى الأنواع الأكثر تعقيدًا.

الرياضيات الأساسية لشبكات GAN معقدة. لحسن الحظ، يمكن للعديد من المقارنات في العالم الحقيقي أن تجعل فهم شبكات GAN أسهل. في المثال السابق، تحدثنا عن مزور فني (مولد) يحاول خداع خبير فني (مميّز). كلما كانت اللوحات المزيفة أكثر إقناعًا، كان على الفنان أن يكون أفضل في التعرف على أصالتها. والعكس صحيح أيضًا: فكلما كان الخبير الفني أفضل في التعرف على أصالة لوحة معينة، كان يجب على المزور أن يعمل بشكل أفضل لتجنب الوقوع في الفخ.

بتعبير أدق، الهدف من المولد هو إنتاج عينات توضح خصائص مجموعة بيانات التدريب بحيث لا يمكن تمييز العينات التي ينتجها عن بيانات التدريب. يمكن اعتبار المولد كنموذج عكسي للكشف عن الكائن. تتعلم خوارزميات التعرف على الكائنات الأنماط الموجودة في الصور للتعرف على محتوى الصورة. بدلاً من التعرف على الأنماط، يتعلم المولد إنشاءها أساسًا من نقطة الصفر. في الواقع، غالبًا ما لا يكون إدخال المولد أكثر من متجه رقم عشوائي. يتعلم المولد من خلال التعليقات التي يتلقاها من المميز. الغرض من المميز هو تحديد ما إذا كان عينة معينة أصليةً (مشتقةً من مجموعة بيانات التدريب) أم مزيفةً (تم إنشاؤها بواسطة المولد). وفقًا لذلك، في كل مرة يتم فيها خداع المميز وتصنيف صورة مزيفة على أنها حقيقية، يعرف المولد أنه قام بعمل جيد. على العكس من ذلك، في كل مرة يرفض فيها المميز بشكل صحيح الصورة التي ينتجها المولد على أنها مزيفة، يتلقى المولد ملاحظات تحتاج إلى التحسين. كما يستمر المميز في التحسن. مثل أي مصنف آخر، يتعلم من الفرق بين تنبؤاته والتسميات الحقيقية (الحقيقية أو المزيفة). وبالتالي، مع تحسن المولد في إنتاج بيانات حقيقية، يتحسن المميز في التمييز بين البيانات المزيفة والبيانات الحقيقية، وتستمر كلتا الشبكتين في التطور في وقت واحد.

فى شبكات التخاصم التوليدية، تحدث ضوضاء فى الشبكة العصبية المولدة التى يتم من خلالها توليد عينات مزيفة، تتمثل مهمة شبكة المميز فى تحديد العينات المزيفة التى تنتجها شبكة المولد. يتم تحديد ذلك من خلال فحص عينات التدريب لمعرفة مدى اختلاف العينة المنتجة عن العينات الفعلية. تعمل هذه الشبكات كعدوين يحاولان التنافس مع بعضهما البعض. فى المراحل المبكرة، يمكن لشبكة المميز بسهولة اكتشاف العينات المزيفة التى ينتجها المولد. ثم تعمل شبكة المولد المنافسة جاهدة

لتقليل الاختلاف بين العينات المزيفة المنتجة والبيانات الحقيقية. إنهم يحاولون إنتاج عينات قريبة من عينات التدريب، مما يجعل هذا تحديًا كبيرًا لشبكة المميز. ومع ذلك، لا تزال شبكة المميز تحاول العثور على زيف البيانات التى تم إنشاؤها. تتنافس كلتا الشبكتين مع بعضهما البعض حتى تجد شبكة المميز صعوبة فى التمييز بين العينات التى تنتجها شبكة المولد والعينات الأصلية.

التدريب على الخصومة

في GAN، يتم تدريب المولد والمميز بطريقة متضاربة، أي أنهما يتنافسان مع بعضهما البعض في إطار المجموع الصفري (**zero-sum**) لإنتاج بيانات مشابهة لتوزيع البيانات الحقيقية. الغرض من المولدفي GAN هو إنتاج عينات يبدو أنها من توزيع بيانات حقيقية، حتى لو كانت مزيفة، والغرض من المميز هو اكتشاف ما إذا كانت العينات مزيفة أم أصلية.

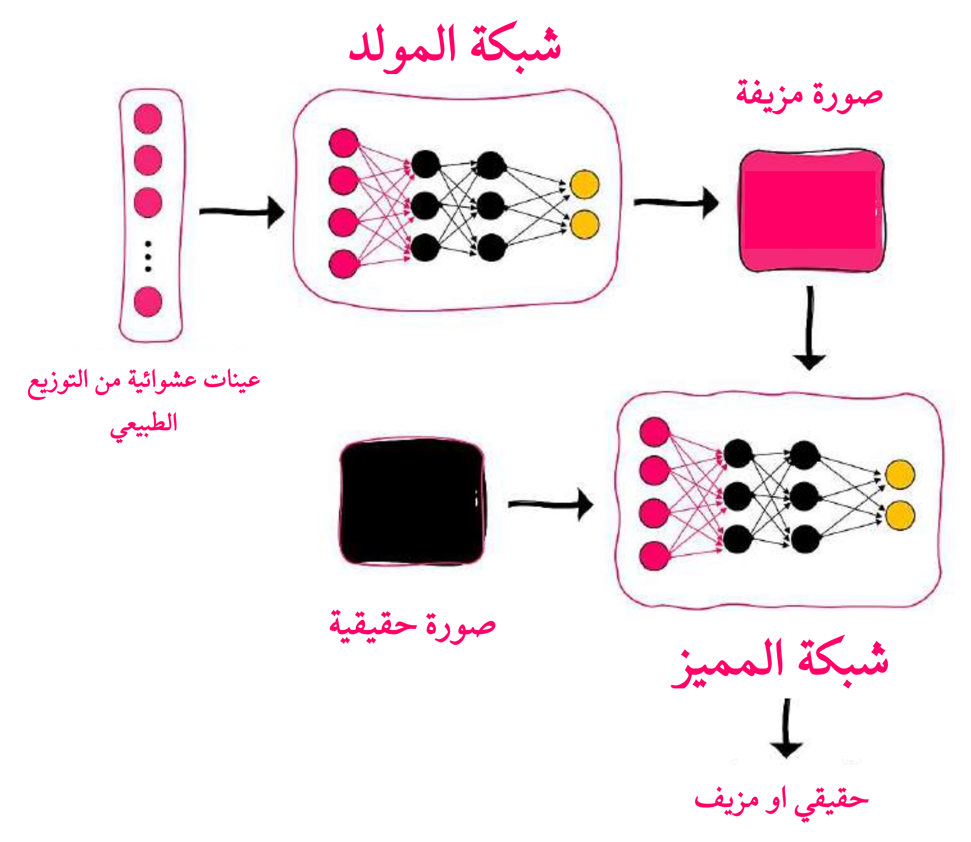
من وجهة نظر التحسين، فإن الهدف من التعليم المولد هو زيادة أخطاء المميز. بمعنى آخر، كلما زاد عدد الأخطاء التي يرتكبها المميز، كان المولد أفضل. الغرض من المميز هو تقليل الخطأ.في كل تكرار، تتحرك كلتا الشبكتين نحو أهدافهما باستخدام تناوب تنازلي. ومن المثير للاهتمام أن كل شبكة تحاول هزيمة شبكة أخرى. يحاول المولد خداع المميز، بينما يحاول المميز ألا ينخدع. أخيرًا، يمكن لبيانات الإخراج (مثل الصور والصوت والفيديو والسلاسل الزمنية) من المولد أن تخدع أكثر المميزات تعقيدًا.

من الناحية العملية، يأخذ المولد عينات عشوائية من توزيع محدد مسبقًا مع التوزيع كمدخلات ويولد البيانات التي يبدو أنها تأتي من التوزيع المستهدف.

يتم تنفيذ الخطوات التالية بواسطة GAN:

1. تأخذ شبكة المولد عينات عشوائية من التوزيع الاحتمالي الطبيعي وتولد الصور.
2. ثم يتم تمرير هذه الصور التي تم إنشاؤها إلى شبكة المميز.
3. تلتقط شبكة المميز كلاً من الصور التي تم إنشاؤها والصور المأخوذة من مجموعة البيانات الفعلية.
4. يعطي المميز الاحتمالات عند الاخراج.
5. يتم حساب الخطأ (دالة التكلفة) للمولد بناءً على الانتروبيا المتقاطعة للصور المزيفة التي يعتبرها المُميز صالحًا.
6. يتم حساب الخطأ (دالة التكلفة) للمميز بناءً على الانتروبيا المتقاطعة للصور المزيفة التي تعتبر مزيفة ، بالإضافة إلى الانتروبيا المتقاطعة للصور الحقيقية التي تعتبر صالحة.
7. في كل فترة ، يتم تحسين كلتا الشبكتين لتقليل اخطائهما.
8. أخيرًا ، ينتج المولد المدرب جيداً الصور كمخرج نهائي يحاكي صور الإدخال الفعلية.

كمثال لتوليد صورة، يمكن عرض نموذج GAN في الشكل التالي:



شكل 6-1. الهيكل العام لشبكة خصومة توليدية.

تدريب GAN

هدفنا الرئيسي هو إنشاء مولدات صور حقيقية (بيانات)، وإطار عمل GAN هو أداة لهذا الغرض. قبل الخوض في مزيد من التفاصيل، دعنا نقدم بعض الرموز:

- نشير إلى المولد بواسطة $G(z, \theta_g)$ ، حيث θ_g هي أوزان الشبكة و z هو المتجه الكامن **(latent vector)** الذي يعمل كمدخل للمولد. فكر في الأمر كبذرة عشوائية (**random seed**) لبدء عملية إنتاج الصور. يحتوي z على توزيع احتمالي لـ $p_z(z)$، والذي يكون عادةً عشوائيًا عاديًا **(random normal)** أو منتظمًا عشوائيًا **(random uniform)**. ينتج المولد عينات وهمية x عن طريق توزيع الاحتمال $p_g(x)$. يمكنك التفكير في $p_g(x)$ كتوزيع احتمالي لبيانات حقيقية فيما يتعلق بالمولد.
- نشير إلى المميز بواسطة $D(x, \theta_d)$، حيث أن θ_d هو وزن الشبكة. يأخذ البيانات الحقيقية كمدخلات عن طريق توزيع $x \sim p_{data}(x)$ أو العينات المولدة $x \sim p_g(x)$. المميز هو

مصنف ثنائي ينتج قيمة 1 أو 0 ، اعتمادًا على ما إذا كانت صورة الإدخال حقيقية (مخرج الشبكة 1) أو تم إنشاؤها (مخرج الشبكة 0)..

- أثناء التدريب ، نشير إلى دوال الخطأ للمميز والمولد بواسطة $J^{(D)}$ و $J^{(G)}$، على التوالي.

يختلف تدريب GAN عن تدريب شبكة عصبية عميقة نموذجية لأن لدينا شبكتين. يمكننا اعتبارها لعبة مجموع صفري(Zero-sum game) كاملة بين لاعبين (مولدين ومميزين):

1. **تسلسلي (Sequential):** هذا يعني أن اللاعبين يتبادلون الأدوار ، على غرار الشطرنج. في البداية ، يحاول المميز تقليل $J^{(D)}$، لكن يمكنه فقط القيام بذلك عن طريق ضبط الأوزان θ_d. يحاول المولد بعد ذلك تقليل $J^{(G)}$، لكن يمكنه فقط ضبط الأوزان ، θ_g. تتكرر هذه العملية عدة مرات.
2. **المجموع_الصفري (Zero-sum):** هذا يعني أن ربح أو خسارة أحد اللاعبين متوازنة تمامًا مع ربح أو خسارة اللاعب الآخر. وهذا يعني أن الخسارة الإجمالية للمولد والمميز هي دائمًا 0:
$$J^{(G)} = -J^{(D)}$$
3. **ميني ماكس (minimax):** هذا يعني أن استراتيجية اللاعب الأول (المولد) هي تقليل الدرجة القصوى للخصم (المميز). عندما نقوم بتدريب المميز ، فإنها تعمل على تحسين اكتشاف العينات الحقيقية والمزيفة (تقليل $J^{(D)}$). بعد ذلك، عندما نقوم بتدريب المولد، فإنه يحاول رفع مستوى المميز المحسن حديثًا (نقوم بتصغير $J^{(G)}$، وهو ما يعادل بتكبير $J^{(D)}$). الشبكتان في منافسة مستمرة. نعرض لعبة **minimax** مع ما يلي أن V دالة تكلفة:
$$\boldsymbol{min_G max_D V(G, D)}$$
لنفترض أنه بعد خطوات تدريب قليلة ، سيكون كل من $J^{(D)}$ و $J^{(G)}$ بالحد الأدنى المحلي. ثم يسمى حل لعبة الحد الادنى **بتوازن ناش (Nash equilibrium)**. يحدث توازن ناش عندما لا يغير أحد الممثلين عمله ، بغض النظر عما يفعله الممثل الآخر. يحدث توازن ناش في إطار عمل GAN عندما يصبح المولد جيدًا جدًا بحيث لم يعد الفارق قادرًا على التمييز بين العينات الفعلية والمُنتجة.

تدريب المميز

إن المميز عبارة عن مصنف شبكة عصبية، ويمكننا تعليمه بالطريقة المعتادة، باستخدام التدرج الاشتقاقي وتدرج الانتشار الخلفي. ومع ذلك، تتكون مجموعة التدريب من أجزاء متساوية من العينات الفعلية والمولدة. دعونا نرى كيف يمكن دمجهافي عملية التدريب:

1. اعتمادًا على نمط الإدخال (حقيقي أو مزيف) ، لدينا مساران:
   - حدد العينة من البيانات الفعلية $x \sim p_{data}$ واستخدمها لإنشاء D_x.

- إنتاج عينة وهمية $x \sim p_g$. هنا ، يعمل المولد والمميز كشبكة واحدة. نبدأ بمتجه عشوائي z، والذي نستخدمه لإنشاء العينة المولدة G_z. ثم نستخدمها كمدخلات للمميز لتوليد الناتج النهائي $D(G(z))$.

2. حساب دالة الخطأ التي تعكس ازدواجية البيانات التدريبية.
3. يتم إعادة نشر تدرج الخطأ وتحديث الأوزان. على الرغم من أن الشبكتين تعملان معًا، إلا أن أوزان المولد θ_g سيتم قفلها ، وسنقوم فقط بتحديث أوزان المميز θ_d. هذا يضمن أننا نقوم بتحسين أداء المميز.

لفهم خطأ التميز، دعونا نتذكر معادلة خطأ الانتروبيا المتقاطعة:

$$CE(p,q) = -\sum_{i=1}^{n} p_i(x) \log(q_i(x))$$

حيث $q_i(x)$ هو احتمال تقدير الناتج الذي ينتمي إلى الفئة i (من فئة n) و $p_i(x)$ هو الاحتمال الحقيقي. للتبسيط، افترض أننا طبقنا المعادلة على عينة التدريب.في حالة التصنيف الثنائي، يمكن تبسيط هذه الصيغة على النحو التالي:

$$CE(p,q) = -(p(x) \log q(x) + (1 - p(x)) \log(1 - q(x))$$

يمكننا تمديد المعادلة إلى مجموعة جزئية من عينات m:

$$CE(p,q) = -\frac{1}{m}\sum_{j=i}^{m}(p(x_j) \log q(x_j) + (1 - p(x_j)) \log(1 - q(x_j))$$

مع العلم بكل هذا دعونا نحدد خطأ المميز:

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}} \log(D(x)) - \frac{1}{2}\mathbb{E}_z \log\left(1 - D(G(z))\right)$$

على الرغم من أنه قد يبدو معقدًا، إلا أنه مجرد خطأ إنتروبيا لمصنف ثنائي مع بعض الحالات الخاصة من شبكات GAN.

تدريب المولد

سنقوم بتدريب المولد على خداع المميز من خلال تحسينه. للقيام بذلك، نحتاج إلى كلتا الشبكتين، على غرار الطريقة التي نعلم بها المميز بأمثلة مزيفة:

1. نبدأ بمتجه كامن عشوائي z ونغذيه من خلال مولد ومميز لإنتاج الناتج $D(G(z))$.

2. دالة الخطأ هي نفس خطأ المميز. ومع ذلك ، فإن هدفنا هنا هو التكبير وليس التقليل. لأننا نريد خداع المميز.
3. في مرحلة التراجع ، يتم قفل أوزان المميز θ_d ويمكننا فقط ضبط θ_g. يتيح لنا ذلك زيادة خطأ المميز إلى الحد الأقصى من خلال تحسين المولد بدلاً من جعل أداة المميز أسوأ.

في هذه المرحلة، نستخدم البيانات التي تم إنشاؤها فقط. سيكون جزء دالة الخطأ الذي يتعامل مع البيانات الحقيقية دائمًا 0. لذلك يمكننا تبسيطها على النحو التالي:

$$J^{(G)} = \mathbb{E}_z \log\left(1 - D\big(G(z)\big)\right)$$

في البداية، عندما يمكن للمميز التمييز بسهولة بين العينات الحقيقية والمزيفة $D\big(G(z)\big) \approx 0$، سيكون التدرج قريبًا من الصفر. هذا يؤدي إلى تعلم طفيف للأوزان، وهو ما يعرف بالتدرج المتناقص (**diminished gradient**). يمكننا حل هذه المشكلة باستخدام دالة خسارة مختلفة:

$$J^{(G)} = -\mathbb{E}_z \log\left(D\big(G(z)\big)\right)$$

لا يزال هذه الخطأ منخفض عندما يكون $D\big(G(z)\big) \approx 1$ وفي نفس الوقت يكون التدرج كبيرًا (عندما يكون أداء المولد ضعيفًا). مع هذا الخطأ، لم تعد اللعبة مجموعة صفرية، ولكن لن يكون لها أي تأثير عملي على إطار عمل GAN.

تدريب الاثنين معا (المولد والمميز)

من خلال معرفتنا الجديدة، يمكننا تحديد هدف الحد الأدنى بشكل كامل:

$$min_G max_D V(G,D) = \frac{1}{2}\mathbb{E}_{x\sim p_{data}} \log\big(D(x)\big) + \frac{1}{2}\mathbb{E}_z \log\left(1 - D\big(G(z)\big)\right)$$

باختصار، يحاول المولد تقليل الهدف، بينما يحاول المميز تكبيره. لاحظ أنه على الرغم من أن المميز يجب أن يقلل من خسارته، فإن الهدف هو تقليل خطأ المميز السلبي. لذلك يجب على المميز تكبيره.

توليد صور جديدة MNIST بـ GAN

في هذا القسم نريد تنفيذ شبكة الخصومة التوليدية باستخدام Keras و Tensorflow. أولاً نقوم بإدخال الاكواد المطلوبة:

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.layers import Input, Dense, Reshape, Flatten, Dropout
from tensorflow.keras.layers import BatchNormalization, Activation, ZeroPadding2D
from tensorflow.keras.layers import LeakyReLU
```

```
from tensorflow.keras.layers import UpSampling2D, Conv2D
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import initializers
import matplotlib.pyplot as plt
import sys
import numpy as np
import tqdm
```

ثم قم بتحميل وتسوية مجموعة بيانات MNIST:

```
(X_train, _), (_, _) = mnist.load_data()
X_train = (X_train.astype(np.float32) - 127.5)/127.5
```

كما لاحظت ، فإننا لا نعيد أي علامات أو مجموعات بيانات اختبار. نحن نستخدم مجموعة بيانات التدريب فقط. العلامات غير مطلوبة ، حيث أن العلامات الوحيدة التي سنستخدمها هي 0 للعلامات المزيفة و 1 للعلامات الحقيقية. هذه صور حقيقية ، لذلك تم تخصيص علامة 1 في المميز لكل منهم.

سنستخدم بيرسيبترون متعدد الطبقات ونعطيه صورة كمتجه مسطح بحجم 784، لذلك سنقوم بتحويل بيانات التدريب:

```
X_train = X_train.reshape(60000, 784)
```

الآن علينا أن نصنع مولدًا ومميزًا. الغرض من المولد هو تلقي مدخلات مشوهة noisy data وإنتاج صورة مشابهة لمجموعة بيانات التدريب. يتم تحديد حجم إدخال الضوضاء بواسطة متغير randomDim. يمكنك تهيئته إلى أي قيمة. عادة ما يتم ضبطه على 100. للتنفيذ ، اخترنا قيمة 10. يدخل هذا الإدخال طبقة كثيفة Dense من 256 خلية عصبية مع تنشيط LeakyReLU. في الخطوة التالية ، نضيف طبقة أخرى متصلة تمامًا بها 512 خلية عصبية مخفية ، تليها الطبقة الثالثة المخفية التي تحتوي على 1024 خلية عصبية ، وأخيرًا طبقة الاخراج مع 784 خلية عصبية. يمكنك تغيير عدد الخلايا العصبية في الطبقات المخفية ورؤية كيف تتغير الدالة. **ومع ذلك، يجب أن يتطابق عدد الخلايا العصبية لكل وحدة إخراج مع عدد البكسل في صور التدريب.** المولد على النحو التالي:

```
randomDim = 10
generator = Sequential()
generator.add(Dense(256, input_dim=randomDim))
generator.add(LeakyReLU(0.2))
generator.add(Dense(512))
generator.add(LeakyReLU(0.2))
generator.add(Dense(1024))
generator.add(LeakyReLU(0.2))
generator.add(Dense(784, activation='tanh'))
```

وبصورة مماثلة ، نقوم بإنشاء المميز. لاحظ أن المميز تأخذ صورًا من مجموعة التدريب أو الصور التي ينتجها المولد ، لذا فإن حجم الإدخال هو 784. ومع ذلك ، فإن الناتج يحدد الى بت واحد،

ويشير 0 إلى صورة مزيفة (تم إنشاؤها بواسطة المولد) ويشير 1 إلى أن الصورة من مجموعة بيانات تدريبية:

```
discriminator = Sequential()
discriminator.add(Dense(1024, input_dim=784,
kernel_initializer=initializers.RandomNormal(stddev=0.02)))
discriminator.add(LeakyReLU(0.2))
discriminator.add(Dropout(0.3))
discriminator.add(Dense(512))
discriminator.add(LeakyReLU(0.2))
discriminator.add(Dropout(0.3))
discriminator.add(Dense(256))
discriminator.add(LeakyReLU(0.2))
discriminator.add(Dropout(0.3))
discriminator.add(Dense(1, activation='sigmoid'))
```

ثم نقوم بدمج المولد والمميز لتشكيل GAN. في GAN ، من خلال تعيين الوسيطة trainable على False ، نتأكد من إصلاح أوزان المميز:

```
# Combined network
discriminator.trainable = False
ganInput = Input(shape=(randomDim,))
x = generator(ganInput)
ganOutput = discriminator(x)
gan = Model(inputs=ganInput, outputs=ganOutput)
```

الحيلة في تدريب الاثنين هي أولاً تدريب المميز بشكل منفصل. نحن نستخدم اخطاء ثنائية عبر الانتروبيا للمميز. بعد ذلك ، نقوم بتجميد freeze أوزان المميز وتدريب GAN الهجين. هذا يؤدي إلى تدريب المولد:

```
discriminator.compile(loss='binary_crossentropy', optimizer=adam)
gan.compile(loss='binary_crossentropy', optimizer=adam)
```

لنبدأ الآن البرنامج التدريبي. لكل دورة ، نأخذ أولاً عينة من noisy data ، ونمررها إلى المولد ، وينتج المولد صورة مزيفة. نقوم بدمج الصور المزيفة وصور التدريب الحقيقية في فئة واحدة مع العلامات الخاصة بنا ونستخدمها لتدريب المميز في الفئة المحددة:

```
dLosses = []
gLosses = []
def train(epochs=1, batchSize=128):
    batchCount = int(X_train.shape[0] / batchSize)
    print ('Epochs:', epochs)
    print ('Batch size:', batchSize)
    print ('Batches per epoch:', batchCount)

    for e in range(1, epochs+1):
        print ('-'*15, 'Epoch %d' % e, '-'*15)
        for _ in range(batchCount):
            # Get a random set of input noise and images
            noise = np.random.normal(0, 1, size=[batchSize, randomDim])
            imageBatch = X_train[np.random.randint(0, X_train.shape[0],
size=batchSize)]

            # Generate fake MNIST images
```

```
            generatedImages = generator.predict(noise)
            # print np.shape(imageBatch), np.shape(generatedImages)
            X = np.concatenate([imageBatch, generatedImages])

            # Labels for generated and real data
            yDis = np.zeros(2*batchSize)
            # One-sided label smoothing
            yDis[:batchSize] = 0.9

            # Train discriminator
            discriminator.trainable = True
            dloss = discriminator.train_on_batch(X, yDis)
```

الآن في نفس حلقة for ، نقوم بتدريب المولد. نريد أن يتم التعرف على الصور التي ينتجها المولد بواسطة المميز على أنها حقيقية ، لذلك نستخدم متجهًا عشوائيًا (ضوضاء) كمدخل للمولد. يقوم بإنشاء صورة مزيفة ثم يقوم بتدريب GAN بطريقة تجعل مميز الصورة يفهم الشيء الحقيقي (الناتج 1):

```
            # Train generator
            noise = np.random.normal(0, 1, size=[batchSize, randomDim])
            yGen = np.ones(batchSize)
            discriminator.trainable = False
            gloss = gan.train_on_batch(noise, yGen)
```

إذا كنت ترغب في ذلك ، يمكنك حفظ اخطاء المولد والمميز وكذلك الصور الناتجة. بعد ذلك، نقوم بتخزين الخطأ لكل دورة وتوليد صور بعد كل 20 دورة:

```
        # Store loss of most recent batch from this epoch
        dLosses.append(dloss)
        gLosses.append(gloss)

        if e == 1 or e % 20 == 0:
            saveGeneratedImages(e)
```

لرسم الاخطاء والصور التي تم إنشاؤها للأرقام المكتوبة بخط اليد، نحدد دالتين مساعدتين ، plotLoss و saveGeneratedImages. اكوادهم على النحو التالي:

```
# Plot the loss from each batch
def plotLoss(epoch):
    plt.figure(figsize=(10, 8))
    plt.plot(dLosses, label='Discriminitive loss')
    plt.plot(gLosses, label='Generative loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.savefig('images/gan_loss_epoch_%d.png' % epoch)

# Create a wall of generated MNIST images
def saveGeneratedImages(epoch, examples=100, dim=(10, 10), figsize=(10,
10)):
    noise = np.random.normal(0, 1, size=[examples, randomDim])
    generatedImages = generator.predict(noise)
    generatedImages = generatedImages.reshape(examples, 28, 28)

    plt.figure(figsize=figsize)
    for i in range(generatedImages.shape[0]):
```

```
        plt.subplot(dim[0], dim[1], i+1)
        plt.imshow(generatedImages[i], interpolation='nearest',
cmap='gray_r')
        plt.axis('off')
    plt.tight_layout()
    plt.savefig('images/gan_generated_image_epoch_%d.png' % epoch)
```

الآن دعونا نحصل على كل الكود معًا:

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.layers import Input, Dense, Reshape, Flatten,
Dropout
from tensorflow.keras.layers import BatchNormalization, Activation,
ZeroPadding2D
from tensorflow.keras.layers import LeakyReLU
from tensorflow.keras.layers import UpSampling2D, Conv2D
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import initializers
import matplotlib.pyplot as plt
import sys
import numpy as np
import tqdm
randomDim = 10

# Load MNIST data
(X_train, _), (_, _) = mnist.load_data()
X_train = (X_train.astype(np.float32) - 127.5)/127.5
X_train = X_train.reshape(60000, 784)

generator = Sequential()
generator.add(Dense(256, input_dim=randomDim)) #,
kernel_initializer=initializers.RandomNormal(stddev=0.02)))
generator.add(LeakyReLU(0.2))
generator.add(Dense(512))
generator.add(LeakyReLU(0.2))
generator.add(Dense(1024))
generator.add(LeakyReLU(0.2))
generator.add(Dense(784, activation='tanh'))
adam = Adam(learning_rate=0.0002, beta_1=0.5)

discriminator = Sequential()
discriminator.add(Dense(1024, input_dim=784,
kernel_initializer=initializers.RandomNormal(stddev=0.02)))
discriminator.add(LeakyReLU(0.2))
discriminator.add(Dropout(0.3))
discriminator.add(Dense(512))
discriminator.add(LeakyReLU(0.2))
discriminator.add(Dropout(0.3))
discriminator.add(Dense(256))
discriminator.add(LeakyReLU(0.2))
discriminator.add(Dropout(0.3))
discriminator.add(Dense(1, activation='sigmoid'))
discriminator.compile(loss='binary_crossentropy', optimizer=adam)
# Combined network
discriminator.trainable = False
ganInput = Input(shape=(randomDim,))
x = generator(ganInput)
```

```
ganOutput = discriminator(x)
gan = Model(inputs=ganInput, outputs=ganOutput)
gan.compile(loss='binary_crossentropy', optimizer=adam)
dLosses = []
gLosses = []
# Plot the loss from each batch
def plotLoss(epoch):
    plt.figure(figsize=(10, 8))
    plt.plot(dLosses, label='Discriminitive loss')
    plt.plot(gLosses, label='Generative loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.savefig('images/gan_loss_epoch_%d.png' % epoch)
# Create a wall of generated MNIST images
def saveGeneratedImages(epoch, examples=100, dim=(10, 10),
figsize=(10, 10)):
    noise = np.random.normal(0, 1, size=[examples, randomDim])
    generatedImages = generator.predict(noise)
    generatedImages = generatedImages.reshape(examples, 28, 28)
    plt.figure(figsize=figsize)
    for i in range(generatedImages.shape[0]):
        plt.subplot(dim[0], dim[1], i+1)
        plt.imshow(generatedImages[i], interpolation='nearest',
cmap='gray_r')
        plt.axis('off')
    plt.tight_layout()
    plt.savefig('images/gan_generated_image_epoch_%d.png' % epoch)
def train(epochs=1, batchSize=128):
    batchCount = int(X_train.shape[0] / batchSize)
    print ('Epochs:', epochs)
    print ('Batch size:', batchSize)
    print ('Batches per epoch:', batchCount)

    for e in range(1, epochs+1):
        print ('-'*15, 'Epoch %d' % e, '-'*15)
        for _ in range(batchCount):
            # Get a random set of input noise and images
            noise = np.random.normal(0, 1, size=[batchSize,
randomDim])
            imageBatch = X_train[np.random.randint(0,
X_train.shape[0], size=batchSize)]

            # Generate fake MNIST images
            generatedImages = generator.predict(noise)
            # print np.shape(imageBatch), np.shape(generatedImages)
            X = np.concatenate([imageBatch, generatedImages])

            # Labels for generated and real data
            yDis = np.zeros(2*batchSize)
            # One-sided label smoothing
            yDis[:batchSize] = 0.9

            # Train discriminator
            discriminator.trainable = True
            dloss = discriminator.train_on_batch(X, yDis)
```

```
            # Train generator
            noise = np.random.normal(0, 1, size=[batchSize,
randomDim])
            yGen = np.ones(batchSize)
            discriminator.trainable = False
            gloss = gan.train_on_batch(noise, yGen)

        # Store loss of most recent batch from this epoch
        dLosses.append(dloss)
        gLosses.append(gloss)

        if e == 1 or e % 20 == 0:
            saveGeneratedImages(e)

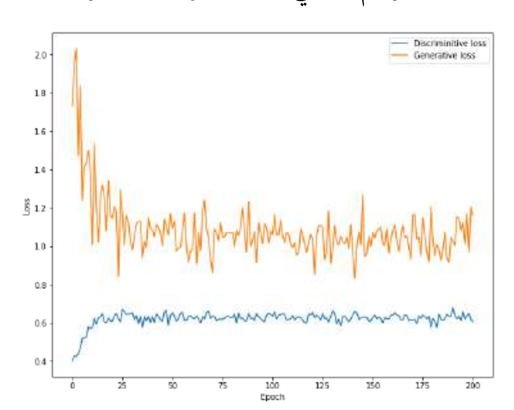
    # Plot losses from every epoch
    plotLoss(e)
```

يمكننا الآن تدريب GAN:

```
train(200, 128)
```

```
Epochs: 200
Batch size: 128
Batches per epoch: 468
--------------- Epoch 1 ---------------
--------------- Epoch 2 ---------------
--------------- Epoch 3 ---------------
--------------- Epoch 4 ---------------
--------------- Epoch 5 ---------------
...
--------------- Epoch 198 ---------------
--------------- Epoch 199 ---------------
--------------- Epoch 200 ---------------
```

في الشكل أدناه ، يمكنك رؤية الرسم البياني لخطأ المولد والمميز لـ GAN الذي يتم تدريبه:



الأرقام المكتوبة بخط اليد التي أنتجتها GAN لدينا في دورات مختلفة هي كما يلي:

Epoch 1: Epoch 20:

Epoch 100: Epoch 200:

كما ترون من الأرقام السابقة ، كلما زاد عدد الدورات التدريبية ، أصبحت أرقام خط اليد التي تنتجها GAN حقيقية أكثر فأكثر.

التحديات المتعلقة بتدريب شبكات الخصومة التوليدية

هناك العديد من التحديات عند العمل مع GAN. قد يكون تدريب شبكة عصبية واحدة أمرًا صعبًا نظرًا للعدد الكبير من الخيارات المتضمنة: **معمارية الشبكة ودوال التنشيط وخوارزمية التحسين ومعدل التعلم ومعدل الحذف العشوائي**. تضاعف شبكات الخصومة التوليدية كل هذه الخيارات وتضيف تعقيدات جديدة. قد ينسى كل من المولد والمميز التكتيكات التي استخدموهافي تدريبهم. يمكن أن يؤدي ذلك إلى وقوع شبكتينفي دائرة من حلول الاستقرار التي لا تتحسن بمرور الوقت. قد تتجاوز إحدى الشبكات الأخرى حتى لا يتعلم أي منهما من الأخرى. أو قد لا

يكتشف المولد الكثير من مساحة الحل الممكنة وسيستخدمها فقط لإيجاد حلول واقعية. تُعرف هذه الحالة باسم **وضع الانهيار** (mode collapse).

يحدث وضع الانهيار عندما يتعلم المولد مجموعة فرعية صغيرة فقط من الحالات الحقيقية المحتملة. على سبيل المثال، إذا كان الأمر يتعلق بإنتاج صور للقطط، فيمكن للمولد أن يتعلم فقط إنتاج صور للقطط السوداء ذات الشعر القصير. يفقد المولد جميع الحالات الأخرى، بمافي ذلك القطط ذات الألوان الأخرى.

تم اقتراح العديد من الاستراتيجيات لمعالجة هذه المشكلة، بمافي ذلك تسوية الدُفعات وإضافة العلامات إلى البيانات التدريبية والمزيد. تؤدي إضافة العلامات إلى البيانات، أي تقسيمها إلى فئات، إلى تحسين أداء شبكات GAN دائمًا تقريبًا. بدلاً من تعلم إنشاء صور للحيوانات الأليفة بشكل عام، يجب أن يكون من الأسهل إنتاج صور، على سبيل المثال، القطط والكلاب والطيور.

خلاصة الفصل

- **بشكل عام ، هناك نوعان رئيسيان من النماذج في التعلم الآلي: نموذج المولد ونموذج المميز.**
- **تركز نماذج المميز على التنبؤ بفئات البيانات وفقًا لخصائصها.**
- **لا يحاول نموذج المولد تعيين سمات للفئات ، ولكنه يولد سمات موجودة في فئة معينة.**
- **تتبع نماذج المولد التعلم غير الخاضع للإشراف الذي يكتشف تلقائيًا الأنماط أو المخالفات في البيانات التي يتم تحليلها.**
- **الغرض الرئيسي من جميع أنواع نماذج المولد هو معرفة التوزيع الفعلي لمجموعة البيانات لمجموعة التدريب.**

اختبار

1. **من منظور الامثلية، ما هو الهدف التدريبي من المولد والمميز؟**
2. **كيف يتم تدريب المميز؟**
3. **ما هي مدخلات المولد؟**
4. **ما هو وضع الانهيار؟**

المراجع

میلاد وزان، (1399). یادگیری عمیق: اصول، مفاهیم و رویکردها، ویرایش نخست، تهران، میعاد اندیشه،

میلاد وزان، (1400). یادگیری ماشین و علم داده: مبانی، مفاهیم، الگوریتم‌ها و ابزارها، ویرایش نخست، تهران، میعاد اندیشه

میلاد وزان، (۱۴۰۰). یادگیری توام برای طبقه‌بندی ویژگی و قطبیت در نظرات فارسی با استفاده از یادگیری عمیق چندوظیفه‌ای، اولین کنفرانس ملی تحول دیجیتال و سیستم های هوشمند، مجتمع آموزش عالی لارستان.

میلاد وزان، (۱۴۰۰). رویکردی جدید برای ارتقا طبقه‌بندی احساسات نقدهای فارسی با استفاده از شبکه عصبی کانولوشنی و طبقه‌بند رای اکثریت، اولین کنفرانس ملی تحول دیجیتال و سیستم های هوشمند، مجتمع آموزش عالی لارستان.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. MIT Press.

Goodfellow, Ian J et al. (2013). “Challenges in representation learning: A report on three machine learning contests”. In: International Conference on Neural Information Processing. Springer

Sutskever, Ilya et al. (2013). On the importance of initialization and momentum in deep learning. In: ICML (3) 28.1139-1147, p. 5.

Siegelmann, H.T. (1995). "Computation Beyond the Turing Limit". Science. 238 (28): 632–637.

Alex Graves and Greg Wayne and Ivo Danihelka (2014). "Neural Turing Machines".

Valentino, Z., Gianmario, S., Daniel, S., & Peter, R. (2017). Python Deep Learning: Next Generation Techniques to Revolutionize Computer Vision, AI, Speech and Data Analysis. Birmingham, UK: Packt Publishing Ltd.

Aljundi, R. (2019). Continual learning in neural networks. arXiv preprint arXiv:1910.02718.

Gupta, P., & Sehgal, N. K. (2021). Introduction to Machine Learning in the Cloud with Python: Concepts and Practices. Springer Nature.

Alpaydin, E. (2004). Introduction To Machine Learning. S.L.: Mit Press.

Bishop, Christopher M (2006). Pattern recognition and machine learning. springer.

Brudfors, M. (2020). Generative Models for Preprocessing of Hospital Brain Scans (Doctoral dissertation, UCL (University College London)).

Charte, F., Rivera, A. J., & Del Jesus, M. J. (2016). Multilabel classification: problem analysis, metrics and techniques. Springer International Publishing.

Chen, Z., & Liu, B. (2018). Lifelong machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 12(3), 1-207.

Dulhare, U. N., Ahmad, K., & Ahmad, K. A. B. (Eds.). (2020). Machine Learning and Big Data: Concepts, Algorithms, Tools and Applications. John Wiley & Sons.

Friedemann Zenke, Ben Poole, and Surya Ganguli. (2017). Continual learning through synaptic intelligence. In International Conference on Machine Learning (ICML).

Gan, Z. (2018). Deep Generative Models for Vision and Language Intelligence (Doctoral dissertation, Duke University).

Jebara, T. (2012). Machine learning: discriminative and generative (Vol. 755). Springer Science & Business Media.

Lesort, T. (2020). Continual Learning: Tackling Catastrophic Forgetting in Deep Neural Networks with Replay Processes. arXiv preprint arXiv:2007.00487

Marsland, S. (2015). Machine Learning : an algorithmic perspective. Boca Raton, Fl: Crc Press.

Mitchell, T.M. (1997). Machine learning. New York: Mcgraw Hill.

Rhys, H. (2020). Machine Learning with R, the tidyverse, and mlr. Simon and Schuster.

Zhou Z. (2021). Machine Learning. Springer Nature Singapore Pte Ltd.

Weidman, S. (2019). Deep Learning from Scratch: Building with Python from First Principles. O'Reilly Media.

Gulli, A., Kapoor, A., & Pal, S. (2019). Deep learning with TensorFlow 2 and Keras: regression, ConvNets, GANs, RNNs, NLP, and more with TensorFlow 2 and the Keras API. Packt Publishing Ltd.

Vasudevan, S. K., Pulari, S. R., & Vasudevan, S. (2022). Deep Learning: A Comprehensive Guide. Chapman and Hall/CRC.

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). Dive into deep learning. arXiv preprint arXiv:2106.11342.

Ekman, M. (2021). Learning Deep Learning: Theory and Practice of Neural Networks, Computer Vision, NLP, and Transformers Using TensorFlow. Addison-Wesley Professional.

Huang, S. C., & Le, T. H. (2021). Principles and labs for deep learning. Elsevier.

Wilmott, P. (2019). Machine learning: an applied mathematics introduction. Panda Ohana Publishing.

Kelleher, J. D., Mac Namee, B., & D'arcy, A. (2020). Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies. MIT press.

Du, K. L., & Swamy, M. N. (2013). Neural networks and statistical learning. Springer Science & Business Media.

Ye, J. C. (2022). Geometry of Deep Learning: A Signal Processing Perspective (Vol. 37). Springer Nature.

Howard, J., & Gugger, S. (2020). Deep Learning for Coders with fastai and PyTorch. O'Reilly Media.

Stevens, E., Antiga, L., & Viehmann, T. (2020). Deep learning with PyTorch. Manning Publications.

Saitoh, K. (2021). Deep Learning from the Basics: Python and Deep Learning: Theory and Implementation. Packt Publishing Ltd.

Ghatak, A. (2019). Deep learning with R (Vol. 245). Singapore: Springer.

Calin, O. (2020). Deep learning architectures. Springer International Publishing.

Sewak, M., Karim, M. R., & Pujari, P. (2018). Practical convolutional neural networks: implement advanced deep learning models using Python. Packt Publishing Ltd.

Liu, Y. H., & Mehta, S. (2019). Hands-On Deep Learning Architectures with Python: Create deep neural networks to solve computational problems using TensorFlow and Keras. Packt Publishing Ltd.

Ma, Y., & Tang, J. (2021). Deep learning on graphs. Cambridge University Press.

Buduma, N., & Locascio, N. (2017). Fundamentals of deep learning: Designing next-generation machine intelligence algorithms. " O'Reilly Media, Inc.".

Ramsundar, B., & Zadeh, R. B. (2018). TensorFlow for deep learning: from linear regression to reinforcement learning. " O'Reilly Media, Inc.".

Hope, T., Resheff, Y. S., & Lieder, I. (2017). Learning tensorflow: A guide to building deep learning systems. " O'Reilly Media, Inc.".

Osinga, D. (2018). Deep learning cookbook: practical recipes to get started quickly. " O'Reilly Media, Inc.".

Trask, A. W. (2019). Grokking deep learning. Simon and Schuster.

Wani, M. A., Bhat, F. A., Afzal, S., & Khan, A. I. (2020). Advances in deep learning. Springer.

Bhardwaj, A., Di, W., & Wei, J. (2018). Deep Learning Essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling. Packt Publishing Ltd.

Gulli, A., & Pal, S. (2017). Deep learning with Keras. Packt Publishing Ltd.

Aggarwal, C. C. (2018). Neural networks and deep learning. Springer, 10, 978-3.

Chollet, F. (2021). Deep learning with Python. Simon and Schuster.

Foster, D. (2019). Generative deep learning: teaching machines to paint, write, compose, and play. O'Reilly Media.

Patterson, J., & Gibson, A. (2017). Deep learning: A practitioner's approach. " O'Reilly Media, Inc.".

Glassner, A. (2018). Deep learning: From basics to practice. Seattle, WA: The Imaginary Institute.

Pointer, I. (2019). Programming PyTorch for Deep Learning: Creating and Deploying Deep Learning Applications. O'Reilly Media.

Deep Learning

From Basics to Building Deep Neural Networks with Python

Milad Vazan

Translated Into Arabic by
Dr. Alaa Taima